7/12/2007: Papers, presentations and video links are now available at Technical Program page.

5/29/2007: NCSA Tours

5/25/2007: Parking information

5/15/2007: Early registration deadline has been extended to May 16th

4/10/2007: Keynote Talk and the Panel are posted.

4/10/2007: Author instructions for camera ready papers are posted.


4/1/2007: Registration Link is added.

NOSSDAV 2007 sessions will be broadcasted live at the following addresses.

- http://media.cs.uiuc.edu/live/NOSSDAV07.asx
- mms://dcs-video1.cs.uiuc.edu/NOSSDAV07

Streaming will be active only during the sessions. For the schedule, please refer to the technical program. Note that the timezone is CDT (UTC/GMT -5 hrs).

Welcome to the NOSSDAV 2007 website. NOSSDAV 2007 will be held in University of Illinois at Urbana-Champaign, IL, USA, June 4-5, 2007.

In its 17th year, the International Workshop on Network and Operating Systems Support for Digital Audio and Video has a long and successful history of bringing together the top researchers, students, developers, and practitioners from academia and industry to discuss new ideas and future directions in multimedia applications, networking, operating systems, and other related areas of computing. Over the years, the scope of NOSSDAV has broadened to include networked games, sensor networks, multimedia interfaces, and peer-to-peer networking. For 2007, we will continue NOSSDAV's emphasis on emerging research areas and fostering extensive discussion of current and future research directions.

This year, a few of accepted NOSSDAV papers with highest quality are selected and their authors are invited to submit an extended version of their papers to a special issue of ACM/Springer Multimedia Systems Journal (MMSJ). MMSJ is a well-known International journal, published by the Springer Verlag, and sponsored by the ACM SIGMM organization. This journal publishes new research results in the areas of multimedia systems, multimedia networking, multimedia applications, multimedia security, multimedia education, multimedia coding, and multimedia retrieval systems.

We are very excited to be continuing the NOSSDAV tradition and look forward to your participation. If you have any questions or comments, please feel free to contact the program co-chairs at the following email addresses:

- **Reza Rejaie**: reza at cs dot uoregon dot edu
- **Klara Nahrstedt**: klara at cs dot uiuc dot edu

**Important Dates**

- **Paper registration**: February 5, 2007
- **Extended Paper submission**: February 19, 2007
- **Notification of acceptance**: March 31, 2007
- **Camera ready version**: April 20, 2007
- **Workshop**: June 4-5, 2007
## Technical Program

### June 4, 2007

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>Chair/Speaker</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:00-</td>
<td>Opening</td>
<td>Reza Rejaie, Klara Nahrstedt</td>
</tr>
<tr>
<td>9:15-</td>
<td>Session 1 - Streaming &amp; Display</td>
<td>Chair: Kevin Almeroth</td>
</tr>
<tr>
<td>9:15-</td>
<td>Mosaicing videos to stream over multiple independent channels</td>
<td>Christopher Boehnen, Allison Regier, Deborah Thomas, Surendar Chandra, Patrick Flynn (University of Notre Dame, US)</td>
</tr>
<tr>
<td>10:45-</td>
<td>Streaming Terrains</td>
<td>Francis Chang, Wu-chi Feng (Portland State University, US)</td>
</tr>
<tr>
<td>10:45-</td>
<td>Blink: Advanced Display Multiplexing for Virtualized Applications</td>
<td>Jacob Hansen (University of Copenhagen, DK)</td>
</tr>
<tr>
<td>11:00-</td>
<td>Keynote Talk: QoS in Wireless Mesh Networks: A Challenging Endeavor</td>
<td>Ralf Steinmetz (Technische Universität Darmstadt)</td>
</tr>
<tr>
<td>11:00-</td>
<td>Lunch Break</td>
<td></td>
</tr>
<tr>
<td>12:00-</td>
<td>Session 2 - Gaming</td>
<td>Chair: Wu-chi Feng</td>
</tr>
<tr>
<td>1:30-</td>
<td>Enhancing Game-Server AI with Distributed Client Computation</td>
<td>John Douceur (Microsoft, US), Jacob Lorch (Microsoft Research, US), Frank Uyeda (University of California, San Diego, US), Randall Wood (Microsoft, US)</td>
</tr>
<tr>
<td>3:00-</td>
<td>RACS: A Referee Anti-Cheat Scheme for P2P Gaming</td>
<td>Steven Webb, Sieteng Soh, W. Lau (Curtin University of Technology, AU)</td>
</tr>
<tr>
<td>3:00-</td>
<td>Traffic Analysis Beyond This World: the Case of Second Life</td>
<td>Stenio Fernandes (CEFET-AL, BR), Carlos Kamienski (University of the ABC, BR), Djamel Sadok (Federal University of Pernambuco, BR), Josilene Moreira (Federal University of Pernambuco, BR), Rafael Antonello (Federal University of Pernambuco, BR)</td>
</tr>
<tr>
<td>3:00-</td>
<td>Break</td>
<td></td>
</tr>
<tr>
<td>3:15-</td>
<td>Session 3 - Coding</td>
<td></td>
</tr>
</tbody>
</table>
**4:45**  
*Chair: Baochun Li*

**PAT: Peer-Assisted Transcoding for Overlay Streaming to Heterogeneous Devices**  
Dongyu Liu (George Mason University, US), Eric Setton (Stanford University, US), Bo Shen (HP Labs, US), Songqing Chen (George Mason University, US)

**Multidimensional Transcoding for Adaptive Video Streaming**  
Jens Brandt, Lars Wolf (TU Braunschweig, IBR, DE)

**Optimal Partitioning of Fine-Grained Scalable Video Streams**  
Mohamed Hefeeda, Cheng-Hsin Hsu (Simon Fraser University, CA)

---

**7:00**  
NOSSDAV Social Events

---

**June 5, 2007**  
**9:00-10:30**  
*Session 4 - IPTV  
Chair: Indranil Gupta*

**When is P2P technology beneficial to IPTV services?**  
Yennun Huang (AT&T Laboratories Research, US), Zhen Xiao (IBM Research, US), Yih-Farn Chen (AT&T Labs - Research, US), Rittwik Jana (AT&T Labs Research, US), Michael Rabinovich (Case Western Reserve University, US), Bin Wei (AT&T Labs - Research, US)

**Performance evaluation of P2P VoIP application**  
Rodrigo Barbosa (Federal University of Pernambuco, BR), Carlos Kamienski (University of the ABC, BR), Denio Mariz (CEFET-PB, BR), Arthur Callado (Federal University of Pernambuco, BR), Stenio Fernandes (CEFET-AL, BR), Djamel Sadok (Federal University of Pernambuco, BR)

**Measuring P2P IPTV Systems**  
Thomas Silverston, Olivier Fourmaux (University Pierre et Marie Curie, FR)

**10:30- Break**

**10:45 - Panel**

**12:15**  
*Topic: Large Scale Peer-to-Peer Streaming & IPTV Technologies  
Panelists:*

- **Aaron Colwell** (RealNetworks)
- **Sugih Jamin** (University of Michigan)
- **Jin Li** (Microsoft Research)
- **Klara Nahrstedt** (UIUC)

*Presentation*

**12:15 - Lunch Break**

**1:30 - 3:00**  
*Session 5 - Measurement  
Chair: Matthias Hollick*

**Evaluating SIP Proxy Server Performance**  
Erich Nahum (IBM Research, US), John Tracey (IBM, US), Charles Wright (IBM Research, US)

**Characterising User Interactivity for Sports Video-on-Demand**  
Andrew Brampton, Andrew MacQuire, Idris Rai, Nicholas Race, Laurent Mathy
(Lancaster University, UK), Michael Fry (The University of Sydney, AU)

**Scalable Application-Specific Measurement Framework For High-performance Network Video**
Congxiao Bao, Xing Li, Jinpeng Jiang (Tsinghua University, CN)

<table>
<thead>
<tr>
<th>Time</th>
<th>Session 6 - Mobility &amp; Middleware</th>
</tr>
</thead>
<tbody>
<tr>
<td>3:00-</td>
<td>Break</td>
</tr>
<tr>
<td>3:15</td>
<td></td>
</tr>
<tr>
<td>3:15-</td>
<td>Location and Mobility in a Sensor Network of Mobile Phones</td>
</tr>
<tr>
<td>4:45</td>
<td>A Middleware for Implementation and Evaluation of Application Layer Multicast Protocols in Real Environments</td>
</tr>
</tbody>
</table>

**Location and Mobility in a Sensor Network of Mobile Phones**
Aman Kansal, Feng Zhao (Microsoft Research, US)

**QoS Adaptation for Realizing Interaction between Virtual and Real Worlds in Pervasive Network Environment**
Shinya Yamamoto (Nara Institute of Science and Technology, JP), Yoshihiro Murata (Nara Institute of Science and Technology, JP), Naoki Shibata (Shiga University, JP), Keiichi Yasumoto (Nara Institute of Science and Technology, JP), Minoru Ito (Nara Institute of Science and Technology, JP)

**A Middleware for Implementation and Evaluation of Application Layer Multicast Protocols in Real Environments**
Kazushi Ikeda, Thilmee Baduge, Takaaki Umedu, Hirozumi Yamaguchi, Teruo Higashino (Osaka University, JP)
**NOSSDAV 2007**

17th International workshop on Network and Operating Systems Support for Digital Audio & Video

**Urbana-Champaign, IL, USA**

**June 4-5, 2007**

---

**How to get to Urbana:** There are different ways to get to Urbana-Champaign.

1. You can **directly** fly to **Urbana-Champaign** Willard Airport in Savoy via American Eagle airline (from Chicago or Dallas) or via Northwest airline(from Detroit).

2. Fly into **Indianapolis** via other airlines (US Air, Continental, etc) and then rent a car and drive 2 hours west on I-74W to Urbana-Champaign.

3. Fly into **Central Illinois Regional Airport** at Bloomington-Normal (via Delta Airline) then rent a car and drive 1 1/2 hours east on I-74E to Urbana-Champaign.

4. Fly into **Chicago** via any airlines, rent a car and drive 2 1/2 hours south on I-190E/I-294S (toll-road)/I-80W/I-57S/I-74E to Urbana-Champaign.

**Hotel Information:**

- We have negotiated with **Hampton Inn** a special rate ($104 plus tax) that you can use by mentioning affiliation with ACM/NOSSDAV 2007 when you make your reservation. The cut-off date to use this rate will be **May 4th, 2007**. To make a reservation, please directly call Hampton Inn Champaign/Urbana (217-337-1100). Remember to use the group code 'CMM'.

- The Hampton Inn and the UIUC Department of Computer Science (Siebel center) are about eight miles from Willard Airport in Savoy, Illinois. The Hampton Inn hotel has a shuttle service. There is a call box at the airport to call the hotel. If your flight arrives after 3:00 p.m. please call ahead to make a reservation to make sure they have a driver on the schedule. The phone number to the hotel to call shuttle is: 217-337-1100 Shuttle runs 8:00am - 9:00pm. There is parking area at the hotel available at no fee.

**Taxi and Rental Car in Urbana-Champaign**

- Yellow Cab 217-355-3553
- The Taxi Company 217-355-1212
- Courtesy Vans 217-359-9134
- Illini Limo 217-384-5892
- National Rental Car 800-227-7368
- National Rental Car at Willard Airport 217-359-5259

**Driving Directions from Willard Airport to Hampton Inn:** From the airport follow the road to the stop light, turn north or left, which is Rt. 45, also known as Neil Street. Continue on Neil Street until you reach downtown Champaign, University Avenue. Turn east or right on University Avenue. Stay on the University Avenue. The last stop light before you reach the hotel is at Wright Street. This street is the division between Champaign and Urbana. Get in the left lane as the Hampton Inn hotel entrance will be on your left. The Hampton Inn hotel is immediately next to the Perkins Restaurant, with entrance off of University Avenue.

**Riding a Bus from Willard Airport:** If you fly into Willard Airport, you can take the 27 **Air Bus** to the Illini Union or the Hampton Inn. If you get off the bus at the Illini Union, you will then need to walk or take a cab about three to four blocks to the Hampton. The
Hampton Inn also has a free airport shuttle with a call box at the airport to call the shuttle. See the Hotel information above.

- **Walking Directions from Hampton Inn to Siebel Center:** Turn left from the Hampton Inn and walk on University Ave until you get to Goodwin Ave (around 200m). Turn right on Goodwin Ave; pass the parking garage and the new NCSA building. The next building after NCSA is Siebel Center (201 North Goodwin Avenue, Urbana).

- **Driving Directions from Hawthorn Hotel to Siebel Center:** If you make a hotel reservation at another hotel such as the Hawthorn hotel, the Hawthorn hotel provides a shuttle to bring you to the Siebel Center. If you choose to drive continue on Neil Street until you reach downtown Champaign, University Avenue. Turn east or right on University Avenue. Stay in the right hand lane. When you reach Goodwin Avenue, turn right. The parking garage is on that corner, so you will make a sharp right after you turn onto Goodwin Avenue. Please park at the meters or you will be towed.

- **Parking Information:** For participants who will drive to NOSSDAV 2007 location, Siebel Center, 201 North Goodwin Avenue, Urbana, and need to park their cars, the following options are available around the Siebel Center: (meters take quarters only)
  - Long-term UIUC parking meters up to 10 hours in the parking garage at the corner of Goodwin/University avenue (across the Hampton Inn). It is in walking distance to the Siebel Center.
  - Long-term UIUC parking meters in the open parking lot, called B-1, which is between Goodwin Avenue and Matthew Street and next to Springfield Avenue. Again the parking is in the walking distance to the Siebel Center.
  - Short-term Urbana parking meters (2 hours) along the Matthew Street (between Springfield and University Ave.), along the Springfield Avenue (between Wright and Goodwin Ave.), and along the Wright Street (between Springfield and University Ave.). All streets are around the Siebel Center in walking distance.

- **Express Bus:** If you fly to Chicago (O'Hare or Midway) instead of Willard, you can take The LincolnLand Express or Suburban Express to Champaign. LincolnLand Express also serves the airports at Bloomington-Normal and Indianapolis. For either, you should make reservations.

- **Rental Cars:** If you rent a car and drive from Chicago, it should take you about 2.5 - 3 hours (150 miles); from Bloomington-Normal IL the drive is about 45 minutes (45 miles), and from Indianapolis it is about 2 hours drive (107 miles). Amtrak also runs regular train service from Chicago to Champaign, but that is somewhat less accessible from the airports.

- **Touring NCSA:**
  - Prof. Nahrstedt has arranged four tours for NOSSDAV '07 participants to visit the National Center for Supercomputing Applications (NCSA) at UIUC. These four 20-minute tours are scheduled on Monday June 4th, between session 3 and the social event and start at 5:00, 5:20, 5:40 and 6:00pm and each can take up to 15 people.
  - Some information about NCSA: "The National Center for Supercomputing Applications (NCSA), one of the five original centers in the National Science Foundation's Supercomputer Centers Program, opened its doors in January 1986. Since then, NCSA has contributed significantly to the birth and growth of the worldwide cyber-infrastructure for science and engineering, operating some of the world's most powerful supercomputers and developing the software infrastructure needed to efficiently use these systems (for example, NCSA Telnet and, in 1993, NCSA Mosaic(tm), the first readily available graphical Web browser). Today the center is recognized as an international leader in deploying robust high-performance computing resources and in working with research communities to develop new computing and software technologies." For further information about NCSA visit [http://www.ncsa.uiuc.edu](http://www.ncsa.uiuc.edu)
  - If you are interested in taking this tour, please send an email to: Jay A Patel <jaypatel@uiuc.edu> (with title "NCSA tour reservation") and provide your name and...
institutions and specify which tour you would like to attend (5:00, 5:20, 5:40, 6:00pm).

- **More information on hotels, restaurants, visitors sites**
  - Urbana-Champaign area can be found at [http://www.visitchampaigncounty.org/](http://www.visitchampaigncounty.org/)

- **Location of the conference at UIUC**
  - Department of Computer Science
    - Thomas M. Siebel Center for Computer Science, 201 N Goodwin Ave
    - Urbana, IL 61801-2302

- **Location of Reception:** TBA

- **Recreational Attractions:**
  - [Allerton Park](#) - 40 minutes drive from Urbana
  - [Lincoln Multimedia Presidential Library](#) - Springfield, IL
  - [Illinois Amish Country](#)
  - [Krannert Museum](#) in Champaign, IL
  - [Spurlock Museum](#) in Urbana, IL
  - [Japan house and gardens](#) - Champaign has a strong tradition in Japanese culture led by a group of Japanese scholars living in Urbana-Champaign and teaching at UIUC.
  - [Dana-Thomas House](#) - in Springfield IL, this is Frank Lloyd Wright architecture (more houses can be found in Chicago). Wright is a very famous architect in Illinois and in USA with very distinct architecture.
  - [Springfield Capitol building](#)

- Information about the Champaign/Urbana Community - restaurants, hotels, entertainment, etc. ([UIUC web site](#))
- Information for Champaign county visitors ([Convention & Visitors Bureau: VisitChampaignCounty.org](#))
- Information about University and the community ([UIUC web site](#))
- General information on Illinois and central Illinois ([Illinois Bureau of Tourism: EnjoyIllinois.com](#))
Keynote Talk

QoS in Wireless Mesh Networks: A Challenging Endeavor

Abstract

Wireless Mesh Networks allow for the self-organizing formation and organic growth of wireless networks. Possible usage scenarios include public community networks, but also provider-operated wireless backbone networks. As a result, mesh networks cater to the vision of a wireless Internet. It is foreseen that such mesh networks are going to change the communication landscape in the domain of wireless networks; the impact might be as high as the impact of the Peer-to-Peer paradigm on the Internet.

However, one crucial success factor is still missing: Quality of Service support. In contrast to wire-line networks, the wireless medium is typically a shared medium. Moreover, in wireless (multihop) networks it is not feasible to address the QoS challenge with the brute-force overprovisioning approach employed in the wire-line domain. As a result, wireless mesh networks are far from adequately supporting multimedia applications such as gaming or voice/video-conversation.

In this talk, we discuss the introduction of QoS mechanisms into the MeSH mode of IEEE 802.16. In particular we discuss compatibility issues with the Point-to-Multipoint mode of 802.16 and show that cross-layer optimization is necessary to achieve the desired performance. Finally, we discuss selected avenues of research that open up on top of the proposed QoS mechanisms.

Biography

Since early 1996, Dr. Ralf Steinmetz has been a professor at the dept. of Electrical Engineering and Information Technology as well as at the dept. Computer Science of the Darmstadt University of Technology, Germany. There he is in charge of a chair position as managing director of the "Multimedia Communications Lab". From late 1996 until late 2001 he directed the Fraunhofer (former GMD) Integrated Publications and Information Institute IPSI. In 1999 he founded the Hessian Telemedia Technology Competence Center (httc e.V.). On whose board he has since served as chair.

His research interests cover networked multimedia issues with the vision of "seamless multimedia communications"; i.e. network dependability and security (e.g. gateways, firewalls), quality of service (e.g. network engineering), content distribution networks (e.g. streaming), context aware communications (e.g. peer-to-peer mechanisms), media semantics (e.g. ontology enrichment, metadata). At Darmstadt he relates these research issues often very closely to mobility, Internet telephony and telemedia learning.
He has been the editor and co-author of a multimedia course, which reflects the major issues of the first (updated in several versions) in-depth technical book on multimedia technology. He has worked as an editor of various IEEE, ACM and other journals. He has served as chair, vice-chair and member of numerous program and steering committees of communications and multimedia workshops and conferences. He is a member of the GI and VDE-ITG. He was awarded as ICCC Governor, the honour of Fellow of both, the IEEE and the ACM. In 2005 he became member of the technology advisory board of the “Hessen Agentur” and he was appointed as the advisor for information und communications technology by the Hessian government.
Author Instructions for Camera Ready Version

This page contains instructions for authors of accepted papers to NOSSDAV'07 on how to prepare and submit the camera ready versions of their papers.

Copyright form

The lead author of each paper should take the following steps to submit the ACM copyright form in order to include the paper in the NOSSDAV proceeding and the ACM digital library.

1. Please make sure that the title and author information on the copyright form match exactly those on the paper itself.
2. Please read the copyright form instructions carefully and only sign the appropriate places (e.g., do not sign both the "government employee" and "non-government employee" part).
3. Please fax the signed copyright form to +1-217-244-6869 and add "Attention: Klara Nahrstedt" on the cover page.

Formatting Information

The final version paper must strictly follow the ACM format as specified below.

1. Please use the ACM style file to prepare the camera ready of your paper as PDF file. Note that we only accept PDF file. Use the Option 2: LaTeX2e - Tighter Alternate style - to reduce the total number of pages.
2. In your latex file, use the following settings:

   \documentclass{sig-alternate}
   \begin{document}
   \conferenceinfo{NOSSDAV}{'07 Urbana, Illinois USA}
   \CopyrightYear{2007}
   \crdata{978-1-59593-746-9/06/2007}

3. If you are using other publishing packages, please follow these instructions.
4. The camera ready versions of your paper should not exceed 6 pages.
5. Do not use page numbers in your document.
6. The graphs of many submitted papers were too small to read, especially the fonts on the axis. No graph should be smaller than a 1/3 of the textwidth.
   Here are some useful tips. In gnuplot you can use

   set pointsize 1.5

   and to increase the pointsizes, you can use

   set term post eps "Helvetica" 18
Also in matlab, you can use

```matlab
set(gca,'TickDir','out')
set(gca,'FontSize',14)
```

to increase font size (and move ticks to the outside of the axis).

## How to Submit

All papers should be submitted **both** via EDAS and via email to jinliang@cs.uiuc.edu (with email subject "NOSSDAV Paper Filename"). Please use the following convention to name the pdf file that you submit via email (and use in the email subject):

`p-<first_author_last_name>.pdf`

For example: p-smith.pdf where hyphen is a separator; and "smith" is the first author's last name, no caps. Use lowercase throughout file names-DO NOT CAPITALIZE. There are rules for special names:

- a hyphenated last name remains hyphenated e.g., Wu Yen-Tse ----> yen-tse
- a compound last name takes an underscore e.g., Robert van Gulik ----> van_gulik
- accents in last names are omitted e.g., Wolfgang Schr<o with umlaut>der ------> schroder (NOT Schroeder) or Jean Ren<e with acute> ------> rene (NOT renee)
- apostrophes in last names are replaced by an underscore e.g., Elizabeth O'Neill -------> o_neill
- Retain periods after abbreviated name elements e.g. Melissa St. Thomas -------> st_thomas
- Drop suffixes from file names e.g. Corbin Jones Jr. -------> jones

## Deadlines

1. Please Fax us the copyright form as soon as possible.
2. Each paper should be associated with one registration at full rate by **April 20** in order to be included in the proceeding.
3. The camera ready version of your paper must be submitted through edas and via email by **April 20**.
There are three registration levels at NOSSDAV 2007. For each level, there is a discount for being an ACM member. The levels are:

- **Full Rate**: Each paper presented at the workshop must be associated with one full rate registration.
- **Affiliated Rate**: Additional non-student authors of an accepted paper or any researcher associated with the author of a paper (at the same institution) are eligible for this rate.
- **Student Rate**: Students are eligible for this discount. Student (co-)authors are eligible for this rate after one full rate registration is associated with their paper.

NOSSDAV 2007 workshop registration fees are presented in the following table:

<table>
<thead>
<tr>
<th>Registration Fee (USD)</th>
<th>Early*</th>
<th>Late/On site</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Member</strong></td>
<td>$415</td>
<td>$465</td>
</tr>
<tr>
<td><strong>Non-member</strong></td>
<td>$440</td>
<td>$490</td>
</tr>
<tr>
<td><strong>Affiliated</strong> with author</td>
<td>$290</td>
<td>$340</td>
</tr>
<tr>
<td><strong>Student</strong></td>
<td>$140</td>
<td>$190</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$165</td>
</tr>
</tbody>
</table>

* Early Registration Deadline: May 14th, 2007  May 16th, 2007

** Member rate is available to ACM/SIG members at the registration time.

[Click here for NOSSDAV 2007 registration](http://www.cs.columbia.edu/~hgs/nossdav/2007/reg.html)
News and Updates

- 7/12/2007: Papers, presentations and video links are now available at Technical Program page.
- 5/29/2007: NCSA Tours
- 5/25/2007: Parking information
- 5/15/2007: Early registration deadline has been extended to May 16th
- 4/10/2007: Keynote Talk and the Panel are posted.
- 4/10/2007: Author instructions for camera ready papers are posted.
- 4/1/2007: Registration Link is added.

Live Webcast

- NOSSDAV 2007 sessions will be broadcasted live at the following addresses.
  - http://media.cs.uiuc.edu/live/NOSSDAV07.asx
  - mms://dcs-video1.cs.uiuc.edu/NOSSDAV07
- Streaming will be active only during the sessions. For the schedule, please refer to the technical program. Note that the timezone is CDT (UTC/GMT -5 hrs).

In its 17th year, the International Workshop on Network and Operating Systems Support for Digital Audio and Video has a long and successful history of bringing together the top researchers, students, developers, and practitioners from academia and industry to discuss new ideas and future directions in multimedia applications, networking, operating systems, and other related areas of computing. Over the years, the scope of NOSSDAV has broadened to include networked games, sensor networks, multimedia interfaces, and peer-to-peer networking. For 2007, we will continue NOSSDAV's emphasis on emerging research areas and fostering extensive discussion of current and future research directions.

This year, a few of accepted NOSSDAV papers with highest quality are selected and their authors are invited to submit an extended version of their papers to a special issue of ACM/Springer Multimedia Systems Journal (MMSJ). MMSJ is a well-known International journal, published by the Springer Verlag, and sponsored by the ACM SIGMM organization. This journal publishes new research results in the areas of multimedia systems, multimedia networking, multimedia applications, multimedia security, multimedia education, multimedia coding, and multimedia retrieval systems.

We are very excited to be continuing the NOSSDAV tradition and look forward to your participation. If you have any questions or comments, please feel free to contact the program co-chairs at the following email addresses:

- **Reza Rejaie**: reza at cs dot uoregon dot edu
- **Klara Nahrstedt**: klara at cs dot uiuc dot edu

### Important Dates

- **Paper registration**: February 5, 2007
- **Extended Paper submission**: February 19, 2007
- **Notification of acceptance**: March 31, 2007
- **Camera ready version**: April 20, 2007
- **Workshop**: June 4-5, 2007
For seventeen years, NOSSDAV has fostered cutting-edge, state-of-the-art research in multimedia and newly emerging areas such as networked games and peer-to-peer streaming. The workshop environment encourages lively discussion among participants and invites strong feedback for work in progress. In 2007, NOSSDAV will be held at the University of Illinois at Urbana-Champaign.

NOSSDAV invites submissions on all areas of multimedia computing and networking and strongly encourages work in progress in emerging areas. Papers grounded in high-quality experimental research based on prototype and real systems are highly valued. Additionally, papers proposing new directions for research or calling into question existing conventional wisdom are welcomed. This year, NOSSDAV will give extra consideration to papers where the source code to experimental or real systems is released. And, NOSSDAV will also give extra consideration to papers that aim to comprehensively validate previous work in some topic within multimedia.

Topics of interest include, but are not limited to:

- Peer-to-peer streaming
- Networked games
- Sensor networks and architectures
- In-network stream processing
- Wireless and mobile multimedia systems
- 3D multimedia and tele-immersion
- Streaming 3D graphics and virtual worlds
- Application-level multicast
- Multimedia security
- Digital rights management
- Real-time operating system support for multimedia
- Multimedia middleware and frameworks
- Multimedia grids
- Programmable and GPU/SPU-enabled multimedia

A broad view will be taken in deciding what topics are within scope. Please feel free to contact the workshop co-chairs if you are unsure and wish to check if a particular paper or topic is within the workshop scope.

As always, student participation is strongly encouraged. To encourage a good mix of seasoned
researchers as well as students, we will be offering discounted registration for non-student co-authors of a paper or other non-student participants associated with the author of a paper (participants from the same institution).

Submissions (as well as the camera ready final versions of accepted papers) should be no longer than 6 pages. We expect these submissions to be the kernel of what will eventually lead to full-length papers at high-quality conferences or journals.

This year, a few of accepted NOSSDAV papers with highest quality are selected and their authors are invited to submit an extended version of their papers to a special issue of ACM/Springer Multimedia Systems Journal (MMSJ). MMSJ is a well-known International journal, published by the Springer Verlag, and sponsored by the ACM SIGMM organization. This journal publishes new research results in the areas of multimedia systems, multimedia networking, multimedia applications, multimedia security, multimedia education, multimedia coding, and multimedia retrieval systems.

**Important Dates:**

- **Paper registration:** February 5, 2007 (5pm PST)
- **Extended Paper Submission:** February 19, 2007 (5pm PST)
- **Notification:** March 31, 2007
- **Camera-ready:** April 20, 2007
- **Workshop:** June 4-5, 2007

**Program Co-chairs:**

- **Reza Rejaie**: reza at cs dot uoregon dot edu
- **Klara Nahrstedt**: klara at cs dot uiuc dot edu
Executive Committee:

- **General Chair:** Klara Nahrstedt - University of Illinois at Urbana-Champaign
- **Technical Program Committee Chair:** Reza Rejaie - University of Oregon
- **Treasurer:** Indranil Gupta - University of Illinois at Urbana-Champaign
- **Publicity Chairs**
  - **Asia:** Wei Tsang Ooi - National University of Singapore
  - **Europe:** Carsten Griwodz - University of Oslo
  - **US:** Dongyan Xu - Purdue University
- **Local Arrangement:** Anda Ohlsson - University of Illinois at Urbana-Champaign
- **Publication Chair:** Jin Liang and Zhenyu Yang - University of Illinois at Urbana-Champaign
- **Web Chair:** Amir Rasti - University of Oregon

Technical Program Committee:

- Tarek Abdelzaher - University of Illinois at Urbana-Champaign
- Kevin Almeroth - University of California, Santa Barbara
- Ehab Al-Shaer - DePaul University
- Ernst Biersack - Institut Eurecom
- Surendar Chandra - University of Notre Dame
- Mark Claypool - Worcester Polytechnic Institute
- Wu-chi Feng - Portland State University
- Carsten Griwodz - University of Oslo
- Indranil Gupta - University of Illinois at Urbana-Champaign
- Kevin Jeffay - University of North Carolina at Chapel Hill
- Ashfaq Khokhar - University of Illinois at Chicago
- Brian Levine - University of Massachusetts Amherst
- Jun Li - University of Oregon
- Baochun Li - University of Toronto
- Ketan Mayer-Patel - University of North Carolina at Chapel Hill
- Liam Murphy - University College Dublin
- Wei Tsang Ooi - National University of Singapore
- Reza Rejaie - University of Oregon
- Pablo Rodriguez - Microsoft Research
- Keith Ross - Polytechnic University
- Henning Schulzrinne - Columbia University
- Prashant Shenoy - University of Massachusetts Amherst
- Shervin Shirzohammadi - University of Ottawa
- Cormac Sreenan - University College Cork
- Dongyan Xu - Purdue University
Visa Information

The Association for Computing Machinery (ACM) does not issue formal invitation letters for visas but it issues a visa support letter. In order to request a letter, you must email the following information to Ashley Cozzi <acozzi.acmorg@atlarge.net> at ACM:

1. Your name and mailing address as they appear on your passport
2. The name of the conference you are attending (NOSSDAV 2007)
3. Your registration confirmation number
4. If you are the author of an accepted paper at NOSSDAV'07, provide the title of your paper
5. A fax number to fax you the support letter.

Please send your requests for visa support letters 6-8 weeks prior to travel plans.
Mosaicing videos to stream over multiple independent channels

Chris Boehnen, Allison Regier, Deborah Thomas, Surendar Chandra and Patrick Flynn
Department of Computer Science and Engineering, University of Notre Dame
Notre Dame, IN 46556, USA
{cboehnen, aregier, dthomas, surendar, flynn}@nd.edu

ABSTRACT

Streaming high fidelity multimedia objects requires large amounts of network bandwidth resources. Sometimes these resources are achieved by aggregating a number of independent and lower capacity network channels. Network level aggregation schemes can stream the single video across all the network links. However, splitting multi-layer encoded video streams are not resilient to network failures on individual links because enhancement layers are not independent and depend on the availability of base layers. In this paper, we investigate several multiple description coding mechanisms that split the stream into multiple independent sub-streams. Our mechanisms attempt to retain the spatial and temporal redundancy inherent in the original stream in order to achieve good compression efficiency. We examine the impact of our approach on changes in peak transmission requirements, overall transmission size and stream quality. We show that the sub-streams are able to sustain substantial data loss while still providing a viewable stream. We also show the object size overhead for the various mechanisms.

Keywords
Multiple Description Coding

1. INTRODUCTION

This work was motivated by application scenarios that operate on high fidelity video streams. The resource requirements of these streams can overwhelm a single communication link. Sometimes, it is preferable to use multiple network links to transmit the stream. For example, remote tele-medicine systems such as Tavarua [7] utilized multiple cellular links to transmit the video streams.

One way to transmit the stream is to use a network level aggregation mechanism that transmits across the multiple links. However, video stream contents are not independent and depend on other parts of the stream. For example, multi-layer encodings depend on the availability of base layers to successfully decode the enhancement layers while MPEG streams require preceding I frames in order to decode subsequent P and B frames. Losing packets on any one of these links can have a catastrophic effect. Though some systems can adapt to lossy networks by reducing the future stream fidelity, transmissions during the actual loss on any of the links can seriously degrade the entire stream. Retransmitting these lost packets on other channels adds latency. In order to stream the media through these independent links, we desire multiple description coding mechanisms (MDC) that can split the original stream into multiple streams, each of which can be adapted to the capacity of the individual channel. We also prefer mechanisms that can continue to operate with the remaining links, albeit at proportionally lower quality levels. The ability to operate through link failures requires MDC mechanisms that create independent set of streams.

In this paper, we explore various ways of splitting a single stream into multiple independent streams. Each of the independent streams can be encoded and transmitted separately. Ideally, each of the sub-streams should add little overall compression overhead. However, modern encoding schemes such as MPEG achieves high compression ratios by exploiting the spatial and temporal redundancy inherent in videos. Unless care is taken in the splitting process, the new sub-streams may have less spatial and temporal redundancy as compared to the original stream, leading to a decrease in compression efficiency (and a corresponding increase in stream size).

In this paper, we explore schemes that retain some of the spatial and temporal redundancy present in the original image. We describe a scheme that maintains spatial redundancy by choosing nearby pixels for the various streams, a scheme that maintains temporal redundancy by choosing nearby frames and a quadrant based approach which maintains relative spatial and temporal redundancy of a smaller portion of the original image. Our experiments show that sub-streams are able to achieve good error resiliency: the received quality proportionally depends on the number of streams that are successfully received.

2. OUR APPROACH

2.1 Objectives

Our goal is to split a stream S to k independent streams s_1, s_2, ..., s_k. Though k can be arbitrarily large, we restrict ourselves to analyzing the behavior of the system using four sub-streams. Ideally, each channel might sport different capacities. For simplicity, we assume that all the channels have equal capacity. Suppose the compressed object sizes of the original and the various sub-streams are N and n_1, n_2, ..., n_k, respectively. We prefer schemes that are:

1. proportional quality: quality of the output stream is proportional to the number of sub-streams that were successfully received. We used PSNR to measure stream quality.
2. space efficient: \( N \leq \sum_{i=1}^{k} n_i \)
3. fair size: equal space requirements for the various sub-streams: \( n_1 = n_2 = n_3 = n_4 \)
We focus our attention on MPEG-4 encoded streams. MPEG-4 compression exploits the spatial redundancy using intra-coded frames (I frames) and temporal redundancy using inter-coded frames (P and B frames). The choice of the number of I, P and B frames depend on the motion characteristics of the particular stream. Streams with high temporal redundancy can achieve high compression ratios using few I frames. However, the loss of inter coded 'I' frames are catastrophic for successfully decoding subsequent P and B frames. Hence, streamed videos incorporate additional 'I' frames in order to achieve good resiliency. We explore mechanisms that create sub-streams that retain some of the spatial and temporal redundancy found in the original stream. Next we outline our proposed mechanisms (Figure 1).

2.1.1 Spatial Separation

The spatial separation mechanism (Figure 1(a)) retains much of the spatial redundancy in the original stream. Each of the four pixels in every 2x2 pixel block of each frame is assigned to a separate stream. Nearby pixels are expected to continue to retain any spatial correlation from the original image (every other pixel in the original stream becomes neighbors in the sub-stream). The sub-streams are spatially reduced by a factor of two on both the dimensions. One advantage of this approach is the easier opportunity for error correction; lost pixels from one or more sub-streams can be estimated through linear interpolation of the neighboring pixels from successfully received streams.

2.1.2 Temporal separation

The next approach (Figure 1(b)) maintains the temporal redundancy in the original stream by assigning each of four consecutive frames (starting from the first frame) to the different sub-streams. The resulting sub-streams retain the spatial dimensions of the original stream while the frame rates are reduced by a factor of four. The I-frames of the resulting compressed sub-streams can be expected to retain their size in the original stream. Retaining the same inter I-frame distance as the original stream has the effect of effectively quadrapling the inter I-frame distance on the original stream. Chakareski [2] used a similar scheme for their analysis.

2.1.3 Quadrant-based approach

The last approach (Figure 1(c)) is a hybrid that retains the temporal and spatial redundancy of the original stream by assigning the four quadrants of each frame of the original to a sub-stream. The resulting sub-streams were spatially scaled by half on each dimension as the original stream while still retaining the original frame rates. If one of the sub-stream is lost, then a whole quadrant of the original stream will be lost. Depending on the temporal characteristics of the original image, each of the sub-streams might retain differing amounts of temporal redundancy. For example, a sub-stream of a newscast might show high motion in the quadrant where it shows a news clip inlay while exhibiting little motion in other sub-streams. Qureshi used a similar scheme in Tavarua [7].

3. EXPERIMENT SETUP

Next, we describe the experiment setup: streams used, evaluation metrics and experimental setup. We discuss our experimental observations in the next section.

3.1 Video clips used

Our dataset consisted of six video clips.

- NDSet: Three clips were acquired through the Computer Vision Research Lab data acquisition process. The clips were captured using an iSight webcam (640x480), Canon camcorder (720x480) and JVC HD camera (1280x720). These clips consisted of a subject sitting in a chair and uttering an unique phrase. There was little spatial movement in the clip. These clips were representative of application scenarios such as tele-medicine and video conferencing wherein the foreground subject might be talking and gesturing against a relatively static background.

- MotorCycleSet: The other three video clips were downloaded from Motorcycle Online. These clips (320x240) show a motorcycle racing sequence. As the camera was rapidly panned to follow the motorcycle these clips exhibited high motion with the background changing constantly.

Figure 2 shows two frames that were captured five frames apart from each of the sequences used in our experiments. The MotorCycleSet frames show high motion as the motorcycle moves at a high speed along the highway.

---

1www.nd.edu/~cvrl/UNDBiometricsDatabase.html
2www.motorcycle.com/mo/mcvideos/videos.html
3. Evaluation architecture

For our experiments (illustrated in Figure 3), we split the video clips into four sub-streams, compressed each of the sub-streams independently, decompressed them and then combined them to reconstruct the original stream. We analyzed the total size of the sub-streams as well as the quality of the reconstructed stream. We experimented with five different encoders that were included with the ffmpeg [1] software package. We used FFmpeg/fldshow ISO MPEG-4 (FMP4), ffmpeg 1/2 (MPG2), Flash Sorenson Video (FLV1), Motion JPEG (MJP), and Sorenson v1 (SVQ1). We specified the target bitrate for the different streams. We observed that the MPEG4 encoder consistently achieved higher compression efficiencies; both for the original stream as well as for the sub-streams. We report our experiences with MPEG-4 encoding in this paper. We forced a staggered start for the various sub-streams (similar to Qureshi et. al. [7]). We experimentally varied the number of frames between I-frames from 3 to 90 frames and plotted the PSNR and compression efficiency in Figure 4. We observed that the streams exhibited high compression efficiency and PSNR for choosing values over 12 frames. Hence, we chose a fixed inter I-frame distance of 12 frames for the rest of the experiments.

3.3 Evaluation Metrics

We used three performance metrics to evaluate our approach. First, we examined the impact of file size and PSNR for transmitting using the sub-streams. We also analyzed the impact of data loss during transmission. In order to have repeatable performance, we manually corrupted some compressed data from the various sub-streams before decompressing and attempting to combine the various sub-streams and recreate the full stream. For our experiments, we simulated data loss of the I-frames. This represents the worst case behavior in terms of data loss; losing P or B frames are expected to be much less disruptive. We determined the file offsets of the various I-Frames and replaced the first 1500 bytes (typical MTU size) with zeros in order to simulate data loss during transmission. We varied the number of lost I-frames among the sub-streams. The modified MPEG4 files were decoded by ffmpeg (using its error compensation mechanisms) before the streams were combined to recreate the original stream. Missing data in our spatial separation mechanism was recreated by using a linear interpolation of surrounding pixels. We also examined the peak transmission requirements of the various streams over time. The magnitude of the peaks represents the burstiness of the different methods.

4. RESULTS

4.1 Resiliency to loss on individual links

The primary motivation behind our sub-stream mechanism was to achieve resiliency for loss over any of the network links. The original MPEG-4 stream has relationship among the various I, B and P frames and so could not independently recover from loss on the different links. For our experiments, we modified the first 1500 bytes (typical MTU size) of the I-frames to zero. Distorting the I-frames has a significant effect on the stream quality because the error propagates to subsequent P and B frames. We performed experiments distorting all or the first half of the I-frames. For the sub-streams, we repeated experiments distorting one, two, three or all the sub-streams. We allowed the decoders to recover from the distortion. For the spatial separation method, we also used our linear interpolation to recover from lost sub-streams. We plot the PSNR values for losing various number of sub-streams among the NDSet and MotorCycleSet in Figure 5. Ideally, we prefer PSNR values that degrade gracefully with increasing data loss.

At a high level, we note that the sub-streams achieve graceful degradation of quality with increasing data loss. For the spatial and the quadrant-based mechanisms, the sub-streams have the same frame rate as the original; distorting each I-frame corresponds to four times the data loss (6000 bytes vs 1500 bytes per I-frame). Still, even with I-frame loss in all the links for half the I-frames, the spatial and quadrant based mechanisms are competitive for ND-set. Similarly, distortion in all I-frames achieves the same quality loss for NDSet and MotorCycleSet. Our linear interpolation significantly improved the PSNR for both the data sets. The temporal separation faced similar data loss as the original. These results were sensitive to compression parameters that are discussed in further detail in Section 4.2.

4.2 Sub-stream characteristics

The previous section showed that our approach can create independent streams that gracefully lost quality on data loss on any of the network links. MPEG-4 streams offer many configurable parameters; our goals are to understand the specific range of parameters. For example, the original stream was compressed for a target of 175 kbps. For our experiments, for each of the sub-streams, we configured the encoder to target streams of bandwidth requirements of 5, 10, 25, 50, 75, 100, 125, 150 and 175 kbps. For each of these streams, we plot the total object size (sum of the four sub-streams) against the stream PSNR (by comparing with the original stream against the recreated stream). We plot the data for NDSet and MotorCycleSet in Figure 6. From Fig-
Figure 4: Effects of varying the inter I-frame distance

(a) PSNR

(b) Video file size

Figure 5: Distortion of I-frame data. PSNR calculated by comparing the original stream against the recreated stream. Note that the initial PSNR of the streams was not the same due to differences in encoding parameters. Hence, the different plots may not be directly compared, even though the relative differences within a separation mechanism are significant.
Figure 6: Combined size and PSNR characteristics of split sub-streams

(a) NDSet DV camera

(b) NDSet high definition camera

(c) MotorCycleSet

Figure 7: Stream size between successive I-frames

Figure 6, we observe that the sub-stream mechanism can add significant overhead, especially when creating a higher quality stream. For example, for the NDSet stream that uses a DV camera (Figure 6(a)), the original, quadrant, temporal and spatial streams consumed about 171 kB, 210 kB, 250 kB and 395 kB of total size to achieve a PSNR value of 33.1, respectively. We noticed similar overhead across all our streams and mechanisms. Sub-streams appear less efficient from a compression perspective. In order to further understand the different sub-stream mechanisms, we plot the total stream size between successive I-frames (GOP size). We plotted the values for the original stream as well as the various individual sub-streams for the NDSet and MotorCycleSet in Figure 7. Ideally, we prefer values for the sub-streams which are a quarter of the original stream. As can be seen from Figure 7, the GOP size of the individual stream depended on the actual stream. Sometimes, the sub-stream GOP size is even larger than the original stream. Future work will investigate mechanisms to predict this behavior so that automatic choices can be developed to choose the appropriate sub-stream mechanism. Also, in Section 4.1, we chose a encoding target rate of 150 kbps for the original stream and 25 kbps for the sub-streams in the NDSet and 175 kbps for original, 50 kbps for spatial and quadrant and 75 kbps for temporal for sub-streams in the MotorCycleSet.

4.3 Peak stream requirement

One of the benefits of splitting the streams is that the sub-streams are expected to have smaller I-frame sizes, especially for the quadrant and spatial separation schemes; temporal separation mechanisms retains the original spatial dimensions and hence can be expected to have similar I-frame size. Systems such as [7] specifically used stream splitting for this purpose. We plot the peak stream
requirements for the NDSet and MotorCycleSet in Figure 8. We observed that I-frames of the sub-streams were relatively smaller. This smoothing effect can also be preferable in scenarios where a stream is split and transmitted over a single link.

5. RELATED WORK

Goyal [5] describe various MDC mechanisms for image, audio and video objects. Chakareski [2] used a MDC scheme that created two descriptions by choosing alternate frames for the sub-stream. Qureshi et al. [7] used quadrant separation mechanism for streaming H.264 encoded streams over multiple cellular links. Kim et al. [6] describe an unbalanced MDC mechanism that provided unequally error protection. In general, earlier work addressed the issue of streaming MPEG over IP networks. Feng et al. [3, 4] and Rexford et al. [8] analyzed the streaming behavior and investigated buffer management mechanisms to smooth the effects of streaming MPEG objects. Wang et al. [9] used reference frames to enhance the streaming performance. Recently, Xu et al. [10, 11] investigated mechanisms to adapt a multi-layer encoding for transmission across multiple independent channels. Our work adds to these systems by investigating MDC mechanisms to stream MPEG-4 video over multiple links.

6. CONCLUSIONS

Using multiple independent and lower capacity links to achieve higher performance is becoming popular. We investigated application level mechanisms to stream MPEG-4 streams over these links. Our experimental analysis showed that our sub-streams lose quality gracefully with a corresponding cost in increase in total transmission requirements. The key challenge is to choose the specific compression parameters and the separation mechanism based on the expected object motion characteristics.

Acknowledgments

This was was supported in part by the U.S. National Science Foundation (CNS-0130839 and CNS-0447671). Regier was supported by a Lilly fellowship.

7. REFERENCES


Streaming Terrains
Francis Chang, Wu-chi Feng
Portland State University
Department of Computer Science
{francis, wuchi}@cs.pdx.edu

ABSTRACT
Streaming computer graphics data is challenging because of the need to retrieve large triangle meshes before any display can begin. This paper proposes and analyzes the benefits of borrowing techniques from lossy image compression to implement a novel technique of progressive terrain rendering for streaming over a network. The goal of the work is to provide a quality-aware framework for 3-D rendering of heightfields.

Keywords: View-dependent progressive mesh, heightfield compression, graphics, streaming

1. INTRODUCTION
Virtual reality systems [active, croquet, gEarth, wWIND, sl] have risen in popularity with readily available high-speed networking and affordable consumer computer graphics processing hardware. However, the deployment of networking hardware has not kept pace with the increasing quality and complexity of visualization data. Even with such advances, the resolution of such visualization can easily consume any additional gains in bandwidth.

There is a need for techniques and algorithms that are aware of both network and rendering constraints because data and viewers are often not co-located. Models should be transmitted in a quality-aware manner that allows data to be sent continuously in a compact form that allows clients to view data with progressively increasing quality. To maximize the user’s experience, the order in which data is transmitted should be dictated by the viewer’s local perception.

In this paper, we focus on the streaming delivery of terrain data for fly-overs. The goals of the system are two-fold. First, distant terrain details and data that are outside the viewer's frustum should be transmitted with a low priority. Second, terrain that is near the viewer should be downloaded to the viewing client with high priority. We propose the application of lossy image compression techniques to represent the height fields for terrain data, allowing the fly-overs to start with low latency, while capturing the essence of the terrain being represented. Using the Grand Canyon terrain data set, we compare the efficacy of our approach with several basic graphics streaming engines. Our results show that we can present the user with a high quality interactive experience with smaller delay.

In the following section, we describe some of the related work. In Section 3, we describe our network model and propose a novel height-field representation and compression algorithm to address the problem of terrain streaming.

2. RELATED WORK
Several systems have been implemented for the streaming of computer graphics data. The networked computer game Second Life [sl] is a massively multiplayer online dynamic virtual world that allows users to explore a large 3 dimensional space, where players can create and exchange virtual items. Objects are described using a primitive constructive solid geometry model. Terrain and map information are sent in 256x256 patches of non-progressive JPEG data. Using the JPEG patches, rendering is performed with a triangle-splitting algorithm based on an exponential distance metric.

From the computer graphics field, a significant amount of work has been done in the field of progressive meshing. Most of the work in this area focuses on arbitrary 3-dimensional meshes, as opposed to specific optimizations for heightfields, which we explore in this paper. Moreover, the viewer’s perspective is not taken into account, resulting in suboptimal viewer-independent streaming algorithms [chen, isenburg, allies].

A network-aware transport protocol has been shown to significantly improve speed and quality of progressive streaming in image data by explicitly modelling packet loss and performing out-of-order data processing [raman]. This approach improves the latency of progressive refinement; however it does not consider prioritization of regions of interest nor 3 dimensional geometry.

For streaming terrain, we can use multi-resolution bitmaps for progressive rendering. [reddy] organizes data in a quad-tree structure, with each node representing a refinement of one-quarter of the space. This approach takes only the viewers location into account when streaming, without considering the visual importance of existing geological features in the data. [tsai] extends this approach by considering terrain complexity and culling patches outside the viewer’s frustum, but does not prioritize information based on viewer distance.

A similar approach streaming terrain approach divides the terrain into square tiles, attempting to pre-cache visible areas around the viewer [poudreux]. This
approach tries to minimize computational complexity for CPU-constrained devices by compiling patches into display lists which can be quickly re-rendered by graphics hardware on successive frames.

[duchaineau] proposes a technique for level of detail management to simplify terrain geometry for real-time rendering. ROAM uses triangle decimation to reduce geometrical complexity by considering the visual impact of rendering additional vertices.

3. ALGORITHMS

Before we describe our proposed approach, we first describe some of the basic assumptions we have regarding the system and network.

3.1 System and Network Assumptions

The basic assumptions we make when modelling our system are that
- Local storage and computing power are large relative to network bandwidth.
- The network is reliable and delivers all packets with minimal latency.

These assumptions are chosen to reflect the goal of our research – to construct an algorithm that can deliver a high-quality 3D reconstruction of a terrain over constrained network infrastructure.

The architectural model we follow is to construct a single server and client. The server stores all the world data and sends it to the client in a quality-aware manner. The client is responsible for rendering the scene and sending viewer updated information to the server.

3.2 Lossless Rendering

To simulate a lossless terrain-streaming method, we constructed an adaptation of the ROAM algorithm [duchaineau]. Normally, this algorithm is used for mesh simplification for real-time rendering. In this application, we repurpose the ROAM vertex creation mechanism for use in the prioritization of data for adaptive network streaming. This algorithm will serve as the baseline comparison case for our experimental work.

The ROAM algorithm divides a landscape into square patches that are represented by progressively refined triangle meshes, allowing finer details to be “aggregated” together when network adaptation becomes necessary.

In the coarsest representation, a ROAM patch is represented by two right angle isosceles triangles. As higher detail is demanded, a triangle may be split into two children triangles, introducing an additional vertex (Figure 1).

This triangle mesh is always constructed in a way to prevent the formation of T-junctions – visual cracks in the mesh, formed when two neighbouring triangles are rendered at incompatible detail levels.

In practice, each ROAM patch is represented in memory by a binary tree, with each node representing a triangular area. Each triangle is in turn represented by two smaller triangles that form the descendents of each node. This data structure is referred to as a binary triangle tree (BTT). The BTT is constructed so that travelling down the branches of the tree represents progressive refinement of the terrain mesh, and hence, additional visual detail that can be presented to the user.

In our implementation, there are two BTTs representing each ROAM patch – one on the server, and one on the client. Initially, the server’s BTT will be fully populated with the full terrain geometry, while the client’s BTT will contain only the coarsest representation. Thus, given infinite resources, the BTT on the client would match that of the server.

Data is sent from the server to the client to populate the client’s BTT - the server constructs a vertex stream to send to the client, based on the viewer’s location and orientation, using a distance-variance metric for vertex prioritization. This is similar to the way standard ROAM implements its progressive refinement.

In our implementation, the variance of all the child vertices is divided by the distance of the node from the viewer to form a score for each vertex in the terrain mesh. All nodes not yet downloaded are placed in a priority queue for streaming to the client. These scores are recalculated per frame to avoid sending late data.

It is important to note that this distance-variance metric differs from the original paper which uses bounding volumes to calculate screen-space rendering error. Our approach enables us to pre-calculate much of the per-frame node prioritization, as well as simplify the visual-weighting estimation. We believe that such changes would be necessary in a practical implementation of streaming ROAM.

3.3 Lossy Rendering

Our proposed approach to stream terrain data is to represent map geometry as a collection of 2-dimensional tiled bitmaps. In this approach, the height-fields that will be rendered are represented as “image” data and compressed using JPEG [jpeg]. Thus, the pixel intensity in the image corresponds to the height at a given location on our map. Because terrain data is fairly smooth (modulo
cliffs), we expect that such a representation will efficiently represent terrain data.

In our implementation, the entire terrain is divided into $64^2$ square bitmaps and compressed using JPEG encoding in progressive mode to allow progressive refinement as data is streamed to the client.

In the simple case, all visible patches are streamed with equal priority. Patches that are outside the viewer’s frustum are not downloaded to the client. We refer to this approach as the jpeg-nopri approach.

In an extension to this algorithm, visible patches are prioritized with respect to their distance from the viewer and the size of the compressed patch (Equation 1).

$$\text{patch importance} = \frac{\text{size of patch}}{\text{distance from viewer}} \quad (1)$$

The proximity of the patch to the viewer is used to determine its visual weight, while the size of the compressed patch is used as a coarse metric to determine the patch’s geometric complexity.

The bandwidth from the server is divided among visible patches in proportion to the score yielded from Equation 1. This prioritization is very similar to that presented in [pouderoux]. However, their algorithm estimates a tile’s viewer independent importance based on its height, whereas our approach approximates visual complexity by its compressed data footprint.

The server overhead for implementing this streaming solution is much smaller than the ROAM-based algorithms introduced in Section 3.1. This is because the calculations for determining priority streaming order are coarser-grained and only require a much simplified understanding of client state.

4. EXPERIMENTATION

We have implemented our system using an OpenGL renderer to simulate various fly-throughs over the terrain, in a 640 x 480 viewport. Example images are shown in Figure 2 and Figure 3. Our simulations are constructed on the framework provided by [turner].

The network streaming is completely simulated in a stand-alone program. The simulation models a network with zero latency and a bandwidth of 56kbps.

The choice of a 56kbps stems from the idea that terrain data should only consist of a portion of a true virtual simulation’s network stream. In a realistic scenario the data stream would include information such as objects, buildings, textures and avatars.

This simulation deals only with terrain geometry. Texture information is not sent. In practice, texture information can be generated procedurally. In such approaches, texture is inferred from the terrain geometry and need not be sent over the network.

The output of the client simulations were captured and compared to a full-detail rendering of the walkthrough, using a PSNR metric.

We make the assumption that the viewer is capable of maintaining a constant 25fps refresh rate. Dividing the available bandwidth by the frame rate gives us an allowance of 280 bytes per frame.

4.1 Simulation Dataset

The simulation dataset used in this experiment was the Grand Canyon dataset from The U.S. Geological Survey (USGS) with processing by Chad McCabe of Microsoft Geography Product Unit [usgs]. The subset used for simulation was based on a 2048x2048 grid with 8-bit heightposts (Figure 4), an area of roughly 15000 km$^2$.

To test our streaming framework, we designed three representative walk-throughs to measure the performance of the various algorithms under different conditions.

The simplest terrain walk-through simulation we use simply crosses the simulated grid diagonally from corner to corner. This crossing is accomplished over 2048 rendered frames.

The second walk-through also traverses the terrain from corner to corner. We augment this walk-through by pausing in the center of the map to rotate the viewer 360 degrees. This requires the streaming system to cope with a changing client orientation.

The third walk-through traverses the grid diagonally while continually panning over the terrain. This is the most demanding of the three walk-throughs, requiring the streaming solution to adapt to a constantly changing viewer location and orientation.
4.2 Baseline Simulations

The initial baseline was constructed assuming the client has full knowledge of the entire map geometry. The simulations were run with a full level of detail. This represents the ideal case.

To represent the worst-case simulation, the entire terrain is represented as a 32³ grid, (1 KB of data). This is the coarsest representation our simulation faces (denoted as plane in Figure 5).

The theoretical best results for the lossy rendering algorithm is given by jpeg-full-95 and jpeg-full-100 in Figure 5, which illustrates a client starting off with the entire jpeg representation, compressed at quality levels of 95 and 100, respectively. This study was performed to determine the quality of the data represented by using a lossy compression method (JPEG). Our experimental results show that the maximum quality that a JPEG-based technique can yield is in the 40-50db range. The difference between a jpeg compressed terrain at 95 and 100 quality is small, but measurable.

When the terrain is compressed with JPEG at quality level 95, the result is a 643,414 byte compressed bitmap. Compressing it with JPEG at 100 quality yields a 1,142,924 byte output, increasing the data size by nearly a factor of two.

As expected, our progressive streaming simulation results fall between the two extremes jpeg-95 and plane.

4.3 Simulation Results (Lossless)

The ROAM-based non-lossy streaming algorithm is illustrated as roam in Figure 5. Our simulation counts each vertex as 4-bytes of data (1 byte for height, 3 bytes for XY positional information). This simulation represents the effect of organizing data in a streaming-friendly manner, without applying any compression. Surprisingly, this yields only a marginal improvement in measured image quality.

For comparison, we have also simulated roammax, which is the same algorithm, but counts each vertex as only 1 byte of data. This value was chosen in accordance with the compression factors given by [alliez]. Surprisingly, this four-fold improvement in compression results in only a marginal increase in image quality. This suggests that at this level, much more bandwidth is needed to improve the quality of the experience rather than clever management of resources.

The ROAM-based streaming techniques result in “popping” artifacts – temporal discontinuities formed by the sudden introduction of a new vertex to the terrain mesh. These artifacts are not captured by our PSNR metric, but may prove distracting to the viewer. The visual impact of these artifacts can be lessened by introducing new vertices using a geomorphing technique to smooth the geometric transition between mesh refinement levels [hoppe].

4.4 Simulation Results (Lossy)

Our algorithm using progressive JPEG patches is reported by jpeg and jpeg-nopri in Figure 5. We use JPEG compression with a quality value of 95, to reflect the high end of JPEG’s useful working range. jpeg-nopri is the case where all JPEG patches are streamed with equal priority if they are visible, while jpeg streams data with network priority given to patches closer to the viewer.

Both by jpeg and jpeg-nopri perform well and are bounded conservatively between the predicted best and worst case simulations. Both algorithms significantly outperform the non-lossy examples we have implemented. Although jpeg-nopri can do better than jpeg when the view frustum mis-predicts the future importance of patches, we can see that the jpeg algorithm usually gives better results.
Figure 5: PSNR simulation results. Frame number is on the X axis. The PSNR for that frame (dB) is on the Y axis. The top graph represents a continuous flythrough (0). The middle graph represents a flythrough with a 360° pan in the midpoint (1). The bottom graph represents a flythrough with a continuous 360° pan (2).
During subjective examination of the rendered output, JPEG “ringing” artifacts are not easily observed – the quality increase in the streaming simulation tends to be fast enough that small inaccuracies are removed before they become too close to the viewer. However, blocking artifacts from neighbouring patches being rendered at different detail levels can be distracting.

The most surprising result is that the complexity/distance prioritized streaming technique performs only marginally better than streaming based solely on visibility. This implies that a high compression rate is more important to the visual quality of the simulation than intelligent prioritization of data. This phenomenon will become more pronounced with larger network latency, due to a less accurate prediction by the prioritization mechanism.

As with the ROAM-based progressive streaming techniques, there are temporal artifacts formed by the sudden progressive refinement of a terrain patch. These problems, as with the ROAM-based algorithms, can be solved by applying a geomorphing technique on newly-refined patches to improve frame-to-frame coherence.

4.5 Future Work

In future work, we plan to examine the benefits of using the JPEG2k compression. JPEG2k has an important property that at low bit-rates, it is able to yield a superior image. We predict that this will result in an improvement in the “ramp up” time for our lossy rendering algorithm.

We will also relax the assumptions used in the design of our algorithm. Currently, we do not perform any geometric simplification between the streamed dataset and the video card. We hope to extend our work to take advantage of LOD algorithms such as ROAM, with explicit understanding of the representation of the data being streamed. At the network layer, a stronger model of packet loss and out-of-order processing can be used, borrowing from ideas in [raman] to further optimize use of the network.

Additional streaming heuristics, such as viewer velocity can also be taken into account to better predict the future relevance of data.

5. CONCLUSION

We have proposed a lossy streaming architecture for the representation of 2-dimensional terrain computer graphics data. This approach has been demonstrated to yield promising results for quality client playback of streaming terrain data. This technique has not yet reached the point of deployability, but our results show the room for potential gains in employing lossy streaming techniques in this domain.

Our experimental results demonstrate the importance of achieving a high data compression ratio in order to provide high-quality streaming terrain. This further underscores the importance of adopting lossy encoding techniques, which can yield much higher compression rates than the non-lossy approaches.

6. ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. NSF CNS-0130344. We would like to thank Cory Ondrejka at Linden Lab for discussing their implementation with us, and the anonymous reviewers of an earlier version of this paper for their suggestions and feedback.

7. REFERENCES

[wwind] NASA World Wind, worldwind.arc.nasa.gov
ABSTRACT
Providing untrusted applications with shared and safe access to modern display hardware is of increasing importance. Our new display system, called Blink, safely multiplexes complex graphical content from multiple untrusted Virtual Machines onto a single Graphics Processing Unit (GPU). Blink does not allow clients to program the GPU directly, but instead provides a virtual processor abstraction which they can program. Blink executes virtual processor programs and controls the GPU on behalf of the client, in a manner that reduces processing and context switching overheads. Blink provides its own stored procedure abstraction for efficient hardware access, but also supports fast emulation of legacy OpenGL programs. To achieve performance and safety, Blink employs just-in-time compilation and simple program inspection.

Categories and Subject Descriptors

General Terms
Performance, Experimentation, Security

Keywords
Graphics, virtualization, hardware acceleration, just-in-time compilation, interpretation

1. INTRODUCTION
In addition to their popularity in data centers, Virtual Machines (VMs) are increasingly deployed on client machines, e.g. to allow for compartmentalization of untrusted software downloaded from the Internet [7], or for ease of management [5]. VM technology is quickly becoming commoditized, and it is conceivable that future desktop operating systems will ship with VM-style containers as a standard feature. While initially users may be willing to put up with seeing multiple desktops in a simple picture-in-picture fashion, over time the demand for more integrated and seamless experiences will grow. Users will expect virtualized applications to blend in nicely, and will want to make use of graphics hardware acceleration, for games, simulations, and video conferencing. Our work attempts to address this need by providing virtual machines with access to the powerful accelerated drawing features of modern display hardware, without compromising the safety guarantees of the VM model.

Compared to other I/O subsystems, the display system is harder to multiplex in a way that is both efficient and safe, especially for demanding applications such as 3D games or full-screen video. This is evidenced by the fact that the major operating systems all provide “direct” avenues of programming the graphics card, largely without operating system involvement, but at the danger of being able to crash the graphics adapter or lock up the entire machine [11]. Because of this danger, untrusted software running inside VMs should not be given direct hardware access, and a layer of indirection between client and hardware is necessary. Such a layer should also provide a hardware-independent abstraction, to allow a VM to run unmodified across different types of graphics adapters. One way of implementing this layer is by letting clients program to a high-level API, and have a trusted display system translate API commands into programming of the actual hardware. The trusted translation
step verifies the safety of each API command, and then forwards commands to the graphics card driver. In this way, applications can be prevented from bypassing address space protection with malicious DMA operations, or exploiting bugs in graphics hardware or driver APIs. All use of hardware resources can be tracked and subjected to a policy that prevents one application from causing starvation, e.g. by consuming all memory on the graphics card.

2. DESIGN AND IMPLEMENTATION

This paper describes Blink, a prototype system for providing untrusted VMs with safe and efficient access to modern display hardware. Blink aims to be as safe, simple and flexible as possible, while at the same time retaining full performance.

The goal of Blink is to serve as the display component of a system where each application is encapsulated inside its own VM. Blink is backwards-compatible with existing software, through emulation layers for X11, kernel framebuffers, or OpenGL, but performance may be enhanced by adjusting applications to bypass these layers and use Blink directly.

The Blink prototype runs on top of the Xen [1] virtual machine monitor. The Blink display server runs as a regular user-application inside a Linux guest VM, using commercially developed device drivers for graphics hardware access. The VM running the Blink server mediates access to the graphics hardware, and clients running inside untrusted VMs talk to this server using a shared-memory protocol, as shown in figure 2.

In the remainder of this section, we first briefly introduce 3D programming using OpenGL, and the challenges faced when trying to make an OpenGL-like abstraction to clients in separate protection domains. From there, we describe the key points of our design, and how they have been implemented in our prototype.

2.1 GPU Programming

Most modern GPU programming is done using APIs such as OpenGL [15] or Direct3D [3]. Our work focuses on OpenGL, a standardized API that is available on most platforms. An OpenGL (GL) program is a sequence of API-calls, of the form `glName(args)`, where `Name` is the name of the called GL command. Some commands modify state such as transformation matrices, lighting and texturing parameters, and some result in immediate drawing to the screen. Drawing commands are enclosed in `glBegin()` and `glEnd()` pairs, e.g. the following GL program draws a triangle:

```gl
glBegin(GL_TRIANGLES);
```

An OpenGL API implementation is often split into two parts. The kernel part provides primitives for accessing graphics card DMA buffers and hardware registers, and the user space part—a GPU-specific shared library known as the Installable Client Driver (ICD)—accesses these primitives to translate GL API calls into programming of the GPU. Thus, the output of an OpenGL program can be re-targeted to new hardware, or to a virtualized abstraction, by replacing the ICD installation files.

2.2 Off-Screen Buffers

In a traditional windowing system, each application owns one or more clearly defined rectangular regions of the screen, and such systems expend much effort clipping drawing operations to correctly fall within these regions. With the advent of faster graphics cards with greater amounts of video memory, a new model has become dominant. Pioneered by the MacOSX “Quartz” composited display system, the new approach is to give each application its own off-screen buffer, a contiguous range of video memory to which the application may draw freely. The display system runs a special compositor process which takes care of painting all of the off-screen buffers to the screen, in back-to-front order. The benefits of this model are simplicity and the support for advanced effects such as translucent or zooming windows. The problem with this model is the great amount of memory required for off-screen buffers, to avoid context-switching back and forth between the server and all clients every time the screen is redrawn. There are also cases where off-screen buffers are likely to contain redundant information, e.g. in the case of a video player. A video player will often decode the current video frame to a texture with the CPU, then use the GPU to display the texture mapped onto a rectangular surface. If the video player is forced to go via an off-screen buffer, the frame will have to be copied twice by the GPU, first when it is drawn to the off-screen buffer, and second when the display is composed by the display system. In our proposed system, the decision of whether and how to use off-screen buffers is left to applications. As we shall see, this flexibility is achieved by letting applications execute code as Stored Procedures (SPs) inside the display server.

2.3 BlinkGL Stored Procedures

Virtual machine systems often exploit existing wire protocols when communicating between a VM and the surrounding host environment. For instance, a remote desktop protocol such as RDP or VNC makes connecting a VM to the display relatively straightforward. VNC and RDP communicate screen updates in terms of pixels and rectangles. Remote display of 3D may be achieved in these systems by rendering the final image server-side [8], though at an additional cost in bandwidth, and with the risk of server CPU or GPU contention.

When wishing to render 3D content locally on the client, the common solution is to serialize rendering commands, into batches of Remote Procedure Calls (RPCs) [2], which are then communicated over the wire protocol. In addition to the cost of communication, this method carries a performance overhead, due to the cost of serializing and de-
serializing command streams. In OpenGL, translation costs can be amortized to some extent by the use of display lists, macro-sequences of GL commands stored in video memory. However, display lists are static and only useful in the parts of the GL program that need not adapt to frequently changing conditions. Blink extends the display list abstraction into more general and flexible BlinkGL stored procedures. Because stored procedures are richer in functionality than display lists, they can handle simple user interactions—e.g. redrawing the mouse cursor or highlighting a pushbutton in response to a mouse rollover—independently of the application VM.

BlinkGL is a super-set of OpenGL. BlinkGL stored procedures run on the CPU—inside the display server—and in addition to GL commands they can also perform simple arithmetic and control operations. Stored procedures are sequences of serialized BlinkGL commands, with each command consisting of an opcode and a set of parameters. A part of the opcode space is reserved for special operations for virtual register copying, arithmetic, or conditional forward-jumps. External state, such as mouse coordinates or window dimensions, can be read into registers with special BlinkGL calls, processed, and the results given as arguments to other BlinkGL calls that take register arguments. The Blink server contains a Just-In-Time (JIT) compiler that converts BlinkGL into native CPU machine code that is invoked during screen redrawing or in response to user input. Because of the simplicity of BlinkGL, JIT compilation is fast, and for GL calls the generated code is of similar quality to the output of a C-compiler. Apart from amortizing translation costs, the use of SPs also has two additional benefits: CPU context switching is greatly reduced because each client does not have to be consulted upon every display update, and in many cases the use of off-screen buffers to hold client areas can be avoided by drawing client areas with SPs on the fly.

2.4 Program Inspection

Some aspects of SP execution require additional checking, e.g. to prevent clients drawing outside of their windows. During JIT compilation, commands and parameters are inspected, and filtered or adjusted according to various policies. This inspection allows the Blink server to weed out unwanted client actions, but also enables global optimizations that exploit advance knowledge of client behavior. If the client is known not to use the Z-buffer for depth-ordering of drawing operations, then the Z-buffer does not need to be cleared before invoking the client’s redraw code, and if the client is not enabling transparency, content covered by the client’s rectangle does not need to be redrawn. During compilation, the safety of each SP command is checked, so that out-of-bounds parameters or illegal command sequences may be detected before the SP is allowed to run.

2.5 Versioned Shared Objects

VMs residing on the same physical host may communicate through shared memory, instead of using wire protocols that are likely to introduce extra data copying, especially problematic for large texture or framebuffer objects. Client VMs communicate with Blink through an array of Versioned Shared Objects (VSOs). A VSO is an in-memory data record containing an object identifier (OID), an object type identifier, a version number, and a list of memory pages containing object data. When Blink receives an update notification from a client VM, it scans through the client’s VSO array, looking for updated objects. When a changed or new object is encountered, Blink performs type-specific processing of object contents, such as JIT compilation of stored procedures, and incorporates any changes in the next display update. Each VM may maintain several VSO arrays, to accommodate the use of multiple OpenGL hardware contexts for different clients within the VM, and care is taken to avoid scanning unmodified arrays. The scan is linear in the number of VSOs in the array, but more elaborate data structures can be envisioned if the scalability of the current approach becomes an issue. Figure 3 shows the first two objects in a VSO array containing a stored procedure and a texture object. Most GL commands are called directly by the JIT’ed machine code, but commands taking pointer arguments are treated specially. For example, the glTexImage2D() command uploads a texture from a main memory address to the graphics card. Blink’s version of the command instead takes a VSO OID, which is then resolved into a memory address during compilation. Texture data gets DMA’ed from application to video memory, and inter-VM copying is avoided.

2.6 Virtualizing Standard OpenGL

Rather than expecting all existing OpenGL software to be rewritten as BlinkGL, Blink also contains a compatibility wrapper which allows unmodified OpenGL software to display via Blink. This wrapper is implemented as a custom, client-side ICD, with the help of an additional BlinkGL command, called glEval(), in the Blink server.

The glEval command invokes an interpreter that understands serialized standard OpenGL and executes it immediately, and by combining client-side driver code with a server-side SP in a producer-consumer pair, it is possible to transparently host unmodified OpenGL software. Like other display systems that compose multiple OpenGL clients to a shared display, the wrapper needs off-screen buffers to avoid flicker or artifacts, and to allow arbitrary effects such as transparent windows. However, this functionality is not supported by the Blink display server. Instead, SPs that are part of the client ICD handle this by executing glEval() in the context of an off-screen buffer, and drawing the off-screen buffer onto a texture during the redraw SP. This way, Blink is able to host unmodified OpenGL applications, and to subject them to arbitrary transformations when composing the screen.
For the code interpreted by `glEval()`, the overhead of full JIT compilation is not justifiable. Instead, Blink implements a specialized interpreter for serialized OpenGL streams. When writing an interpreter, the choice is basically between implementing a ‘‘giant-switch’’ with a case for each opcode number, or the ‘‘threaded code’’ approach, with a jump-table with an entry for each opcode number, both of which may be implemented as portable C code. On modern architectures, both approaches suffer from poor branch prediction accuracy, as low as 2%-50%, due to a large number of indirect branches [10]. Furthermore, the ratio of arguments to opcodes is high for serialized OpenGL, so a traditional interpreter has to spend much of its time copying arguments from the input program to the parameter stack of the called GL API function. As a more efficient alternative, we designed a new interpretation technique which we refer to as Stack Replay.

Stack Replay is a simple and fast interpretation technique designed specifically for the characteristics of OpenGL and similar sequences of batched remote procedure calls. It is based on the observation that a serialized OpenGL program is little more than an array of call-stacks. This means that parameter copying can be avoided altogether by pointing the stack pointer directly to opcode parameters before each API call. Branch prediction can be improved by using a minimal code-generator which converts input programs into machine code sequences of API calls interleaved with increments of the stack pointer register, so that arguments are consumed directly from the input program without copying. This approach is platform-specific, but does offer better performance than platform neutral alternatives such as a giant-switch interpreter, which could still be provided as a fallback on other platforms. The platform-specific parts of the interpreter consist of a seven line main-loop in C, and 10 lines of inline x86 assembly pre- and postambles, used when calling a batch of generated code.

When using Linux as the guest VM, the Blink client driver also supports displaying the Linux kernel framebuffer and X11 on an OpenGL texture. This is accomplished by adding a framebuffer driver, which maps a main memory buffer onto a BlinkGL texture, to the guest Linux kernel. Using this driver it is possible to run legacy text mode and X11 applications on top of Blink. Figure 4 shows X11 running on top of a BlinkGL texture.

2.7 Display State Recovery

The precursor to Blink was the 2D Tahoma [7] Display System (TDS). TDS was completely stateless, with the display merely acting as a cache of client content. Among other benefits, this allowed for checkpointing and migration [6] of VMs. BlinkGL clients can be implemented to be stateless, e.g. the server just calls the client’s initialization SP to recreate any missing state, but the transparently virtualized OpenGL programs are not stateless out of the box, because OpenGL is itself a stateful protocol. To solve this problem, we have added a state-tracking facility to the client-side OpenGL emulation layer, so that copies of all relevant state are maintained inside the address space of the calling application. Display lists and textures are captured verbatim, and the effects of transformation to the OpenGL matrix stack are tracked, so that they can later be replayed. Our approach here is in many ways similar to the one described by Buck et.al. [4], but with the purpose of being able to recreate lost state, rather than as a performance optimization.

3. Evaluation

In this section we attempt to measure key aspects of Blink’s performance. At the micro-level we measure the overheads introduced by JIT compilation and interpretation, and at the macro-level we measure overall system throughput for a simple application. These benchmarks do not claim to be an exhaustive evaluation, but they give a good indication of how Blink performs relative to native OpenGL code.

We evaluated Blink on a low-end desktop PC, a Dell Optiplex GX260 PC with a 2GHz single-threaded Intel Pentium4 CPU, with 512kB cache and 1024MB SDRAM. The machine was equipped with an ATI Radeon 9600SE 4xAGP card with 128MB DDR RAM, using ATI’s proprietary OpenGL display driver.

We first evaluated JIT compiler performance. We instrumented the compiler to read the CPU time stamp counter before and after compilation, and report average number of CPU cycles spent per input BlinkGL instruction. Because we did not measure OpenGL performance in this test, all call-instructions emitted by the compiler point to a dummy function which simply returns. We measured instructions spent per executed virtual instruction, and report per virtual instruction averages.

As input we created two programs; the first (OpenGL-mix) is the setup phase of the GLGears application, repeated six times. This program performs various setup operations, followed by upload of a large amount of vertexes for the gear objects with the `glVertex()` command. The second (Arith-mix) is mix of 8 arithmetic operations over two virtual registers, repeated for every combination of the 32 virtual registers. Both programs consist of roughly 8K instructions, performance figures are in table 1. We noticed that subsequent invocations (numbers in parentheses) of the compiler
were almost twice as fast as the first one, most likely because of the warmer caches on the second run. We expect larger programs and multiple SPs compiled in sequence to see similar performance gains. Arithmetic operations on virtual floating point registers are costlier than GL calls, as we make little attempt to optimize them. Finally, we measure the cost of interpretation. We call the interpreter with an input program similar to OpenGl-mix, and measure the cost of interpretation using Stack Replay. As before, all calls are to dummy functions that simply return immediately. We see that the use of interpretation adds about 44% overhead compared with the execution of JIT compiled code (59 vs. 41 cpi respectively). Thus there is benefit to using the JIT compiler in cases where the cost can be amortized, but the interpreter yields reasonable performance as well.

To validate our claim that for GL-calls the JIT compiler produces code of similar-quality as gcc, we also ran the OpenGl-mix code in two scenarios—statically compiled into the display server, and in the JIT compiled version. Performance of the two programs is nearly identical, as can be seen in table 2.

Secondly we measured overall system throughput. For this we ported the classic GLGears demo to display using Blink Stored Procedures. GLGears displays three spinning gears 3D rendered into a 512x512 window. We ran multiple GLGears instances, each in a separate VM, and measured the average time deltas between client updates.

The Blink version of GLGears uses SP register arithmetic to transform the gear objects independently of the client. To gauge the improvement obtained by not having to contact the client for each display update, we run two versions of Gears: GearsSP which uses stored procedure arithmetic and avoids context switching, and GearsSwitch which is updated by the client for each screen redraw. Figure 5 shows time-deltas as a function of the number of VMs. We see that GearsSP is able to maintain a steady frame rate of 50 frames per second for more than three times the amount of VMs than GearsSwitch, due to its avoidance of CPU context switches.

Table 1: Cycles-per-instruction for the JIT compiled SPs, and for interpreted OpenGL streams, on a 2GHz Pentium 4. Numbers in parentheses with warm cache.

<table>
<thead>
<tr>
<th>Type of input</th>
<th>#Instr.</th>
<th>Compile</th>
<th>Execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenGL JIT</td>
<td>8,034</td>
<td>102 (41) cpi</td>
<td>41 cpi</td>
</tr>
<tr>
<td>Arithmetic JIT</td>
<td>8,192</td>
<td>99 (55) cpi</td>
<td>50 cpi</td>
</tr>
<tr>
<td>OpenGL interpr.</td>
<td>8,191</td>
<td>0 cpi</td>
<td>59 cpi</td>
</tr>
</tbody>
</table>

Table 2: Cycles per OpenGL command, when executed using the native driver, or through JIT compiled code.

<table>
<thead>
<tr>
<th>Type of input</th>
<th>Redraw rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Native ATI driver call (gcc 3.3.6)</td>
<td>6564 fps</td>
</tr>
<tr>
<td>Blink Stored Procedure</td>
<td>554 cpi</td>
</tr>
<tr>
<td>Interpreted OpenGL on Blink</td>
<td>552 cpi</td>
</tr>
</tbody>
</table>

Table 3: Frames-per-second, for a 512x512 GLGears demo, on an Intel Core Duo 2, with a 256MB nVidia GeForce 7900GS graphics card.

The final test was run on more modern hardware, a 2.1GHz Intel Core Duo 2 CPU, equipped with a 256MB nVidia GeForce 7900GS graphics card, and nVidia’s proprietary graphics driver. Again, we are using the GLGears OpenGL demo as our benchmark, measuring total frames per second. Results are tabulated in table 3. In line with our previous results, BlinkGL stored procedures execute at close to native speed, while the overhead of interpreting command streams and off-screen buffer rendering results in approximately 25% drop in frames per second, for the OpenGL emulation case. We attribute most of this overhead to the extra copying needed for the off-screen buffer.

4. RELATED WORK

The X Window System (X11) facilitates remote display, and the protocol’s extensibility means that other protocols such as OpenGL can be tunneled inside X connections. Unfortunately, currently popular versions of X have very large code bases, making them hard to trust security-wise. Efforts to create trusted X implementations [9] have not had lasting impact, and many of the assumptions on which X is based (e.g. the need for remote font servers or support for monochrome or color-mapped displays) are no longer relevant. For these reasons, we have chosen not to base our work on X.

Recently, the VMGL [16] project has adopted Chromium [14] to work across VM boundaries, over TCP/IP. Like Blink, VMGL supports state tracking for use in VM checkpointing and migration, and VMGL currently implements a larger subset of OpenGL than Blink. Blink employs JIT compilation and static verification of stored procedures, and saves the overhead of passing all data through a pair of network protocol stacks by optimizing for the common case of client and server being on the same physical machine. VMWare Inc. has also announced experimental support for Direct3D virtualization in their desktop products. Blink does not support Direct3D, but a potential workaround is to run a Direct3D emulator on top of OpenGL, e.g. Transgaming’s Cedega technology.

Specialized secure 2D display systems for microkernels have been described for L4 [12] and EROS [18]. Both systems make use of shared memory graphics buffers between client and server, and both describe mechanisms for secure

http://www.transgaming.com
labeling of window contents. Our work addresses the growing need for 3D acceleration but currently our labeling mechanism is rather crude.

5. FUTURE WORK

Blink allows an untrusted application to drive a 3D display, and allows the application to be implemented in a stateless manner. We are currently working on combining Blink with the self-migration and checkpointing mechanism described in previous work [13]. This combination allows us to live-checkpoint Blink VMs to stable storage, e.g. to a hard- or flash drive, and to fork and live-migrate application VMs across the network. This is similar to the system proposed by Potter et.al. [17], but with the difference that checkpointing is performed from within the VM, and may be done in the background without interrupting the user.

6. CONCLUSION

Blink demonstrates that today’s advanced display devices can be multiplexed in a safe manner without poor performance. Blink emphasizes achieving safety with high performance, since enforcement overhead is often the main obstacle to the adoption of security mechanisms. In particular, a less efficient enforcement mechanism might not scale with the currently rapid growth in GPU capabilities.

Blink achieves safety by using a simple, fast JIT compiler and a shared-memory protocol that also helps reduce the cost of client/server communication. Blink further reduces overhead by amortizing JIT translation costs over multiple display updates and composing multiple applications to the same screen without a need for off-screen buffers. In addition, Blink remains backwards-compatible by employing an efficient and novel batch RPC interpretation technique, known as Stack Replay. As a result, Blink allows for safe, practical display sharing even between soft-realtime graphics applications and legacy code.

7. ACKNOWLEDGMENTS

The author wishes to thank Úlfar Erlingsson for his help in preparing this paper, and Jørgen S. Hansen, Eske Christiansen, Eric Jul, Niraj Tolia, and the anonymous reviewers, for their suggestions and comments on previous versions.

8. REFERENCES


QoS in Wireless Mesh Networks:
A Challenging Endeavor

Keynote Talk
NOSSDAV 2007
17th International Workshop on Network and Operating Systems Support for Digital Audio & Video
Urbana-Champaign, IL, USA
June 4–5, 2007

Prof. Dr.-Ing. Ralf Steinmetz
Dr.-Ing. Matthias Hollick, M.Tech. Parag Mogre

KOM - Multimedia Communications Lab
Dept. of Electrical Engineering and Information Technology, Dept. of Computer Science
TUD – Technische Universität Darmstadt
Merckstr. 25, D-64283 Darmstadt, Germany.
Ralf.Steinmetz@KOM.tu-darmstadt.de Tel.+49 6151 166151, Fax. +49 6151 166152

httc - Hessian Telemedia Technology Competence-Center e.V
Merckstr. 25, D-64283 Darmstadt, Ralf.Steinmetz@httc.de

Overview

1 Basics: The Wireless Network Range
   1.1 It is all about Business
   1.2 Feasibility of Today’s Wireless Architectures?
   1.3 Need for QoS: A Look at Contemporary WMNs
   1.4 Application Scenario Depends on Who is Viewing the WMN
   1.5 QoS Provisioning in WMNs a Holistic Approach
2 State-of-the-art standards for supporting broadband WMNs
   2.1 IEEE 802.16 Standardization
   2.2 QoS at 803.16 / Somehow included
   2.3 802.16 QoS: PMP Mode / Point to MultiPoint Mode at MAC Layer
   2.4 802.16 QoS: MeSH Mode
3 Selected Solutions for Tackling the QoS Challenges in WMNs
   3.1 QoS Architecture for 802.16 MeSH Mode
   3.2 802.16 Distributed Scheduling: Example
   3.3 802.16 MeSH Election
   3.4 Link Level Bandwidth Reservation to Support QoS
   3.5 QoS Architecture for 802.16 MeSH Mode
   3.6 End-to-end Delay Characteristics: A Closer Look
   3.7 Additional Issues
4 Conclusion
1 Basics: The Wireless Network Range

(Geographical) Scale of Network

- **Scale**: Body, Inhouse, Rural Area, Metropolitan Area, Country
- **Paradigm**: WPAN, WLAN, WWAN
- **Technology**: Bluetooth, 802.15, HIPERLAN/2, 802.11, GSM, UMTS

Fill the GAP - why?

1.1 It is all about Business

What You Could Get for a €1 in Early 2005

**Processing**
- One PC-day of CPU time

**Storage**
- 2 GB disk storage
  - > 30h MP3 in 128kbps, i.e. ~ 3 DivX movies

**Interconnection**
- 400 MB broadband data
  - 6 hours of music
- 20 MB ISDN voice tel.
  - 45 minutes talk time
- 0.8 KB SMS
  - 5 messages

Bits ≠ Value

- Wired
  - Broadband: 0.25¢ per MB
- Wireless
  - GPRS: > €1 per MB
  - SMS: €1250 per MB

It’s the Bandwidth/Spectrum that’s expensive
(also missing competition for some services)
Wireless Broadband Access: Vision?

Today:

Day after tomorrow:

Which network architecture will enable cost benefits?

What about Wiring the Last Mile?

The Last Mile:
- Connection between a home and local hub

Challenges:
- Scale & legacy make last mile expensive

Problems esp. in countries other than Germany
- E.g. US
  - ~ 135 million housing units in the US (U.S. Census Bureau 2001)
  - POTS (legacy) network designed for voice & built over 60 years
  - Cable TV networks built over last 25 years
  - Touching each home incurs cost:
    - customer equipment; installation & servicing
- Developing countries
  - large majority of developing world does not have broadband connectivity
  - It is not economically feasible to provide wired connectivity to these customers

Future
- is about rich multimedia services and information exchange
  - ...possible only with wide-scale availability of broadband Internet access
- P2P
- Ubiquitous computing / communicating sensors
Organic Growth for Broadband Wireless Access BWA Coverage

Example Kaiserslautern:
- DSL service is available for only below half of the residents in Kaiserslautern (July 2004)
- Broadband wireless access
  - as alternative to VDSL
    - FTTN (Fibre To The Neighborhood)
    - FTTC (Fibre To The Curb)
    - FTTB (Fibre To The Basement)
    - FTTH (Fibre To The Home).

Example New York:
- In fall 2002 we count 13,707 access points in Manhattan
- 91% are located below 96th street
- Digital Divide!

The provider perspective
- Today strictly hierarchical and tree-structured

1.2 Feasibility of Today’s Wireless Architectures?

The provider perspective
- Today strictly hierarchical and tree-structured
We modeled mobility and workload in a metropolitan area: (figure shows an aerial view of Darmstadt Germany)

Average traffic density (in kByte/s) over the modeled day
Model Predictions – Selected User Densities

The residential areas surround the city center

- Active Residents, 20:00h-21:00h
- Consumer, 00:00h-24:00h
- Trainees, 00:00h-24:00h
- Office workers, 00:00h-24:00h

Wireless Mesh Networks are a Promising Option!

Why Mesh?
- Depart from strictly hierarchical topologies
- Mesh networks to cut down infrastructure costs

Requirements?
- network is able to integrate heterogeneous set of access technologies
  - (e.g. various types of hot spots or local area network technologies)
- There is no central instance for network control and resource management of entire radio access network RAN
  - → decentralized mechanisms
- loose or tight coupling to 3G/B3G environments should be possible

Solutions?
- Using standardized technologies is the key to reduced costs
1.3 Need for QoS: A Look at Contemporary WMNs

Infrastructure Based

SkyPilot, Flarion, Motorola (Canopy)
Invisible Networks, RoamAD, Vivato,
Arraycomm, Malibu Networks,
BeamReach Networks, NextNet
Wireless, Navini Networks, etc.

Infrastructure-less

Meshnetworks Inc., Radiant Networks,
Invisible Networks, FHP, Green Packet Inc.,
LocustWorld, etc.
and e.g. millennial.net

Architecture effects design decisions on
- Capacity management, fairness, addressing & routing, mobility management,
- energy management, service levels, integration with the Internet, etc.

Interoperability?

Source MS-Research

What’s Missing?

Today’s mesh networks
(Source: www.howstrange.com)
Next. gen. mesh networks

However, proprietary solutions
do not scale and are hence not cost effective
Standardized support for QoS at the MAC layer (working crosslayer)
- is the key!
current efforts:
- IEEE 802.16 MeSH mode, IEEE 802.11s proposal
What QoS do We Need to Support in WMNs?

QoS support depends on the applications considered.

Applications

- Rabbits
- Mice
- Sugarcane
- Carrots
- Cheese

QoS requirements

Mesh base station

Mesh subscriber station

1.4 Application Scenario Depends on Who is Viewing the WMN

Application Scenario Depends on Who is Viewing the WMN

End user's perspective

Operator's perspective

Researcher's perspective

[Image Source: Washington CEO Inc.]
Operator’s Perspective: Applications

Delivering Data, Voice and Video to the Home over a Converged Delivery Network

The End-user Perspective: Novel Applications

(Source: MS-Research)
The (some) Companies’ Perspective

Source: Benchmarking Mission

- visit of
  - diverse university labs and
  - companies
- in the area of sensor networks
- with German government
- In May 2007
The Researcher’s Perspective

Communication Networks III - Mobile Networking
EXAM
Term 2007
Multiple Choice

Mesh networks are interesting for researchers because …

- A) they are the solution to all troubles.
- B) several challenges are as yet unsolved before their full potential can be reached.
- C) they provide ideal infrastructure for multiplayer games.
- D) they have potential to allow for organic network growth, which translates into reduced infrastructure cost.
- E) they have potential for becoming ubiquitous.

Answer: B, D, E + C (if “Time” and “Advisor” permits)

Summary of Application Scenarios

WMNs looked upon as
- replacement for DSL

Many community WMNs will involve
- P2P applications and services
- P2P multimedia distribution

Many applications need hard QoS
- Emergency services
- Multiplayer games
- Voice and video conferencing applications
- Distributed file system and backup applications

Mesh allows YOU
- to play a central role in providing services and media content
1.5 QoS Provisioning in WMNs a Holistic Approach

The ring designed for QoS should rule all the following aspects

Applications & Transport
MAC
PHY
Routing
No QoS possible without security and reliability

Mechanisms which work at each layer and across layers are needed

Problem and Solution Space for WMNs

Currently a lot of proprietary solutions exist
• but
  • Do not lead to economies of scale
  • Incompatible solutions, interworking not easy

A standardized solution for (metropolitan) mesh networks?
• 802.16 family (operator perspective)
• 802.11 family (802.11s, end-user perspective)
• But, which technology to choose?

Issues to be considered
• Range and capacity
• Security, privacy, and fairness
• Ease of use
• Self management & self healing
• …

This talk focuses on IEEE 802.16 standard and its MeSH mode.
2 State-of-the-art standards for supporting broadband WMNs

1 Basics: The Wireless Network Range
   1.1 It is all about Business
   1.2 Feasibility of Today’s Wireless Architectures?
   1.3 Need for QoS: A Look at Contemporary WMNs
   1.4 Application Scenario Depends on Who is Viewing the WMN
   1.5 QoS Provisioning in WMNs a Holistic Approach

2 State-of-the-art standards for supporting broadband WMNs
   2.1 IEEE 802.16 Standardization
   2.2 QoS at 803.16 / Somehow included
   2.3 802.16 QoS: PMP Mode / Point to MultiPoint Mode at MAC Layer
   2.4 802.16 QoS: MeSH Mode

3 Selected Solutions for Tackling the QoS Challenges in WMNs
   3.1 QoS Architecture for 802.16 MeSH Mode
   3.2 802.16 Distributed Scheduling: Example
   3.3 802.16 MeSH Election
   3.4 Link Level Bandwidth Reservation to Support QoS
   3.5 QoS Architecture for 802.16 MeSH Mode
   3.6 End-to-end Delay Characteristics : A Closer Look
   3.7 Additional Issues

4 Conclusion

2.1 IEEE 802.16 Standardization

[Diagram: Evolution of 802.16 PHY standard]
IEEE 802.16 Modes of Operation

Point-to-multipoint mode (PMP)

- all subscriber stations (SS) are within range of the base station (BS)

$\rightarrow$ single hop only

Mesh mode (MeSH)$\rightarrow$ Multi hop

Scope of 802.16-2004

Figure 1—IEEE Std 802.16 protocol layering, showing SAPs and e.g. network management not within the scope

(Source 802.16-2004)
2.2 QoS at 802.16 / Somehow included

IEEE 802.16 at least supports QoS
  • PMP vs. MeSH

See e.g.

Technische Universität Darmstadt
Department of Electrical Engineering and Information Technology
Department of Computer Science (Adjoint Professor)
Multimedia Communication Lab
Prof. Dr.-Ing. Ralf Steinmetz

The IEEE 802.16-2004 MeSH Mode Explained
Technical Report
KOM-TR-2006-08

By
Parag S. Mogre, Matthias Hollick, Ralf Steinmetz

Contact: parag.mogre@mei.tuwdr.de
http://www.kom.tu-darmstadt.de

What Does the IEEE 802.16 MeSH Specification Provide?

What we expected
What we found!

Add muscle & meat

What we did!

Page 31 IEEE 802.16-2004:
“Mechanisms are defined in the protocol to allow vendors to optimize system performance by using different combinations of these bandwidth allocation techniques while maintaining consistent interoperability definitions”

→ To enable vendors to place their own algorithms
QoS in PMP mode
- Connection-based / per flow
- DSx messages
  - DSA, DSC, DSD
    - (add, change, delete)
- Service classes
  - UGS - unsolicited grant service
    - real-time, fixed size, periodic basis (e.g. VoIP, Gaming)
  - rtPS - real-time polling service
    - real-time, variable size, periodic basis (e.g. MPEG)
  - nrtPS - non-real-time polling service
    - non-real-time (e.g. high bandwidth FTP)
  - BE - best effort service
    - no QoS guarantee (e.g. HTTP, P2P file sharing)

But: “The connection-based QoS provisioning using the DSx messages are not used” in MeSH mode (chapter 6.3.9.14.10)
→ MeSH mode needs to deal with DSx?
→ What are drawbacks/benefits?
But multihop, no ONE central Basestation available, packet-based

QoS Realization
- On a packet by packet basis
- QoS can only be specified for unicast messages
  - ..., mesh Connection IDentifier CID Identifies one link only

Incompatible to the PMP mode of operation!
- PMP operates on a connection basis
Link Level Bandwidth Reservation to Support QoS

802.16 Scheduling

Point to MultiPoint (PMP)

MeSH Mode

Centralized Scheduling

Distributed Scheduling

Contention-free (Step1) polling by BS (Step2) Request/Grant

Contention-based Mcast/Bcast polling

Contention-based (Collision free, using Control Subchannel)

Coordinated (Collision free, using Control Subchannel)

Uncoordinated (May collide, using Data Subchannel)

What is the Scope of Different Scheduling Schemes for the MeSH Mode?

Legends & Sample Topology

Radio range

Logical link (wired/c)

Subscriber Station (SS)

Base Station (BS) or MeSH BS

Simple Node ID (for explanation only)

Node index (for scheduling)

Node ID (16 bit)

(Source: TU Darmstadt, KOM)
Transmission Opportunities
- fair access
- mesh election

Minislots – unit of reservation

Reservation of Minislots for Transmission

- Nodes store state for each minislot
  - Available
  - Receive available
  - Transmit available
  - Unavailable

- Short-term and long-term reservation possible
  - Short-term: certain no. of frames or
  - Long-term: all future frames – to be actively cancelled
3 Selected Solutions for Tackling the QoS Challenges in WMNs

- UGS: Unspecified Grant Service
- EP: Explicitly Polling Service
- nPS: Non-Explicitly Polling Service
- BE: Best Effort Service

4 Types of Scheduling Service

How?

minislots

4 minislots

1 minislot

frames

4 frames
good until canceled

3.1 QoS Architecture for 802.16 MeSH Mode

Network Layer (NET)
- CS SAP
- MAC SAP

Medium Access Control Layer (MAC)
- Bandwidth Management Module
- MAC Common Part Sublayer (MAC CPS)
- MAC Layer Coordination Function (MLCF)
- Distributed MAC Management Module
- Centralized MAC Management Module

PHY Layer (PHY)
- Control Subframe
- Data Subframe

Legend:
- Interface
- Queue
- PDU/SDU Flow
- Internal Control Flow
- Parameters
- PDU/SDUs
QoS Architecture for 802.16 MeSH Mode

We proposed a QoS Architecture for 802.16 MeSH mode
• Architecture has been designed, implemented, tested
• Results are promising
• Standard-compliant
• Compatible with PMP mode

Compatibility with the PMP mode
• QoS mapping
  • Service Class → CID/Drop Precedence/Reliability

<table>
<thead>
<tr>
<th>Network Layer Priority (e.g., IP Type of Service)</th>
<th>802.16 Service Class</th>
<th>802.16 MSH CID Priority/Class</th>
<th>802.16 MSH CID Drop Precedence</th>
<th>802.16 MSH CID Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>BE</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>non-realtime Polling Service (nrtPS)</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>real-time Polling Service (rtPS)</td>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Unsolicited Grant Service (UGS)</td>
<td>6</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

QoS Architecture for 802.16 MeSH Mode

Uses distributed scheduling to support QoS
• not dependent on the presence of a mesh base station
• centralized scheduling is limited in applicability (will be seen later)

Uses unique combination of the specified classes of distributed scheduling
• considers aspects introduced by the distributed mesh election
• considers requirements of the differing scheduling services
• and at the same time manages bandwidth efficiently

To understand design principles
• a detailed look at the mechanisms in the standard is necessary
3.2 802.16 Distributed Scheduling: Example

Reservation of minislots
Nodes store state for each minislot
- Available
- Receive available
- Transmit available
- Unavailable

802.16 Distributed Scheduling
3-Way-Handshake:
- Request
  - Demand in minislots
  - Persistence (=duration)
  - Slot intervals available
- Grant
  - Slots
  - Start frame number
  - Persistence
3-Way-Handshake:

- **Request**
  - Demand in minislots
  - Persistence (=duration)
  - Slot intervals available

- **Grant**
  - Slots
  - Start frame number
  - Persistence

Grant Confirm
- **Grant Confirmation**
  - Slots
  - Start frame number
  - Persistence
802.16 Distributed Scheduling

3-Way-Handshake:
- **Request**
  - Demand in minislots
  - Persistence (=duration)
  - Slot intervals available
- **Grant**
  - Slots
  - Start frame number
  - Persistence
- **Grant Confirmation**
  - Slots
  - Start frame number
  - Persistence
3.3 802.16 MeSH Election

Principle of MeSH election

- To concurrently access transmission opportunities
- See Cao et. al for analysis

(Source TU Darmstadt)

3.4 Link Level Bandwidth Reservation to Support QoS

802.16 Scheduling

Point to MultiPoint (PMP)  MeSH Mode

Centralized Scheduling  Distributed Scheduling

Contention-free (Step1 polling by BS, Step2 Request/Grant)

Contention-based (NavScan scanning)

Coordinated (Collision free, using Control Subchannel)

Coordinated (Collision free, using Data Subchannel)

Uncoordinated (May collide, using Data Subchannel)
Reservation of Minislots for Transmission Revisited

Reservation of minislots

Short-term and long-term reservation possible
3.5 QoS Architecture for 802.16 MeSH Mode

Simulation study
- Traffic for classes
  - UGS, rtPS, BE
  - Variable packet sizes
    - (100 ... 1000 Byte)
- PHY:
  - QPSK ½, QAM64 2/3
  - Bandwidth 3.5MHz
  - Oversampling 8/7
- MAC:
  - Frame Duration 10ms, 98 Minislots per frame
- Routing:
  - DSR
- Predictor variables:
  - different reservation styles/service classes
- Simulation time 300s

(a) Simulation Topology

Efficient BW usage, high reservation overhead, high e2e delay
In-efficient BW usage, low reservation overhead, low e2e delay
BW-Overhead-Delay trade-off can be controlled

(3) Selected Results

<table>
<thead>
<tr>
<th>Reservation Style/ Architecture</th>
<th>Priority Class</th>
<th># SlotFrames Available</th>
<th># SlotFrames Receive avail</th>
<th># SlotFrames Transmit avail</th>
<th># SlotFrames Unavailable</th>
<th>Average End to-end Delay</th>
<th>Avg. Handshake Delay</th>
<th>Average # of Handshakes per Node (mean over 300)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact Reservation</td>
<td>UG</td>
<td>0.53</td>
<td>1.30</td>
<td>1.37</td>
<td>6.88</td>
<td>0.31</td>
<td>0.42</td>
<td>52.12 sec</td>
</tr>
<tr>
<td></td>
<td>rtPS</td>
<td>2.11</td>
<td>2.47</td>
<td>2.72</td>
<td>11.8</td>
<td>2.10</td>
<td>2.20</td>
<td>57.32 sec</td>
</tr>
<tr>
<td></td>
<td>BE</td>
<td>2.11</td>
<td>2.47</td>
<td>2.72</td>
<td>11.8</td>
<td>2.10</td>
<td>2.20</td>
<td>57.32 sec</td>
</tr>
<tr>
<td>Good until cancelled</td>
<td>UG</td>
<td>0.53</td>
<td>1.30</td>
<td>1.37</td>
<td>6.88</td>
<td>0.31</td>
<td>0.42</td>
<td>52.12 sec</td>
</tr>
<tr>
<td></td>
<td>rtPS</td>
<td>2.11</td>
<td>2.47</td>
<td>2.72</td>
<td>11.8</td>
<td>2.10</td>
<td>2.20</td>
<td>57.32 sec</td>
</tr>
<tr>
<td></td>
<td>BE</td>
<td>2.11</td>
<td>2.47</td>
<td>2.72</td>
<td>11.8</td>
<td>2.10</td>
<td>2.20</td>
<td>57.32 sec</td>
</tr>
<tr>
<td>Quick Architecture</td>
<td>UG</td>
<td>0.53</td>
<td>1.30</td>
<td>1.37</td>
<td>6.88</td>
<td>0.31</td>
<td>0.42</td>
<td>52.12 sec</td>
</tr>
<tr>
<td></td>
<td>rtPS</td>
<td>2.11</td>
<td>2.47</td>
<td>2.72</td>
<td>11.8</td>
<td>2.10</td>
<td>2.20</td>
<td>57.32 sec</td>
</tr>
<tr>
<td></td>
<td>BE</td>
<td>2.11</td>
<td>2.47</td>
<td>2.72</td>
<td>11.8</td>
<td>2.10</td>
<td>2.20</td>
<td>57.32 sec</td>
</tr>
</tbody>
</table>

3.6 End-to-end Delay Characteristics: A Closer Look

Initial delay peaks
Flows start sending data
Scheduling Revisited

Legend & Sample Topology

- Radio range
- Subscriber Station (SS)
- Logical link (wireless)
- Simple Node ID (for explanation only)
- Node ID (16 bit)
- Node index (for scheduling)
- Base Station (BS) or McSH BS

(Source: TU Darmstadt, KOM)

Analysis of Coordinated Centralized Scheduling

And e.g. the larger the control subframe, the smaller the data subframe → less throughput

(Source of some pictures 802.16-2004)
### 3.7 Additional Issues

When designing algorithms critical parameters present in the standard need to be critically analyzed.

- For example:
  - Cao et al. [1] analysis of the mesh election algorithm
  - Our work [2] analysis of the network entry process and implications for QoS
  - Ignoring these factors when designing algorithms may lead to suboptimal performance

Additionally security and end-to-end routing need to be considered

- See e.g. this talk
- And we developed routing algorithms
  - Results are promising

Next steps

- Optimization of the designed algorithms
- Optimizing the interactions across layers
- Much more to come …

### 4 Conclusion

802.16 MeSH Mode …

- is a candidate standard for mesh networks
- lacks adequate QoS and security support for the envisioned applications/scenarios

Our contributions

- Analysis of important features of the standard
- Various bug-fixes, extensions, and algorithms to make the standard work
- Design and analysis of a QoS architecture for 802.16 MeSH
- Analysis of existing Security architecture and design of extensions for MeSH mode

Next Steps

- More to come …
List of References

[...] www.kom.tu-darmstadt.de/publications/ . .
various publications


Additional Reading

The MeSH mode explained


Standardization + Marketing Links 802.16

- http://grouper.ieee.org/groups/802/16/
  - IEEE 802.16 committee
- http://standards.ieee.org/getieee802/
  - Download of selected specifications (see also download area on our course webpage)
- http://www.wimaxforum.org (WiMAX Alliance)

Companies (Software + Hardware)

- http://research.microsoft.com/mesh/
- http://www.meshnetworks.com

NOSSDAV 2007 Program

NOSSDAV 2007

17th International workshop on Network and Operating Systems Support for Digital Audio & Video

Urbana-Champaign, IL, USA

June 4-5, 2007

June 4, 2007

9:00-9:15 Opening
Reza Rejaie, Klara Nahrstedt

9:15-10:45 Session 1 - Streaming & Display
Chair: Kevin Almeroth

Mosaicing videos to stream over multiple independent channels
Christopher Boehnen, Allison Reger, Deborah Thomas, Surendar Chandra, Patrick Flynn (University of Notre Dame, US)

Streaming Terrains
Francis Chang, Wu-Wei Feng (Portland State University, US)

Blink: Advanced Display Multiplexing for Virtualized Applications
Jacobs Hansen (University of Copenhagen, DK)

10:45-11:00 Break

11:00-12:00 Keynote Talk:
QoS in Wireless Mesh Networks: A Challenging Endeavor
Ralf Steinmetz (Technische Universität Darmstadt)
Talk abstract and Biography
NOSSDAV 2007 Program

1:30-3:00  Session 2 - Gaming
Chair: Wu-chi Feng

Enhancing Game-Server AI with Distributed Client Computation
John Douceur (Microsoft, US), Jacob Lorch (Microsoft Research, US), Frank Uyeda (University of California, San Diego, US), Randall Wood (Microsoft, US)

RACS: A Referee Anti-Cheat Scheme for P2P Gaming
Steven Webb, Sieleng Soh, W. Lau (Curtin University of Technology, AU)

Traffic Analysis Beyond This World: the Case of Second Life
Stenio Fernandes (CEFET-AL, BR), Carlos Kamiencki (University of the ABC, BR), Djamel Sadoz (Federal University of Pernambuco, BR), Josielle Moreira (Federal University of Pernambuco, BR), Rafael Antonello (Federal University of Pernambuco, BR)

3:00-3:15  Break

3:15-4:45  Session 3 - Coding
Chair: Bingchun Li

PAT: Peer-Assisted Transcoding for Overlay Streaming to Heterogeneous Devices
Dongyu Liu (George Mason University, US), Eric Setton (Stanford University, US), Bo Shen (HP Labs, US), Songgang Chen (George Mason University, US)

Multidimensional Transcoding for Adaptive Video Streaming
Jens Brandt, Lars Wolf (TU Braunschweig, DE)

Optimal Partitioning of Fine-Grained Scalable Video Streams
Mohamed Hefeeda, Cheng-Hsian Hsu (Simon Fraser University, CA)

---

NOSSDAV 2007 Program

June 5, 2007

9:00-10:30  Session 4 - IPTV
Chair: Jitendar Gaipta

When is P2P technology beneficial to IPTV services?
Yennun Huang (AT&T Laboratories Research, US), Zhen Xiao (IBM Research, US), Yi-Fang Chen (AT&T Labs - Research, US), Ritula Jana (AT&T Labs Research, US), Michael Rabrenovich (Case Western Reserve University, US), Bin Wei (AT&T Labs - Research, US)

Performance evaluation of P2P VoIP application
Rodrigo Barroso (Federal University of Pernambuco, BR), Carlos Kamiencki (University of the ABC, BR), Denio Marz (CEFET-PB, BR), Arthur Callado (Federal University of Pernambuco, BR), Stenio Fernandes (CEFET-AL, BR), Djamel Sadoz (Federal University of Pernambuco, BR)

Measuring P2P IPTV Systems
Thomas Silverston, OlivierFourmaux (University Pierre et Marie Curie, FR)

10:30-10:45  Break

10:45-12:15  Panel
Topic: Large Scale Peer-to-Peer Streaming & IPTV Technologies
Panellists:
- Aaron Colwell (RealNetworks)
- Sugih Jamin (University of Michigan)
- Jin Li (Microsoft Research)
- Klara Nahrstedt (UIUC)

Moderator: Reza Rejaie
NOSSDAV 2007 Program

1:30-3:00  Session 5 - Measurement
Chair: Matthias Hoëck

- Evaluating SIP Proxy Server Performance

- Characterising User Interactivity for Sports Video-on-Demand
  Andrew Brempton, Andrew MacQuire, Idris Rai, Nicholas Race, Laurent Mathy (Lancaster University, UK), Michael Fry (The University of Sydney, AU)

- Scalable Application-Specific Measurement Framework For High-performance Network Video
  Congxiaox Bao, Xing Li, Jirjeng Jang (Tsinghua University, CN)

3:00-3:15  Break

3:15-4:45  Session 6 - Mobility & Middleware
Chair: Sumanar Chandhi

- Location and Mobility in a Sensor Network of Mobile Phones
  Aman Kansal, Feng Zhao (Microsoft Research, US)

- QoS Adaptation for Realizing Interaction between Virtual and Real Worlds in Pervasive Network Environment
  Shinpei Yamamoto (Nara Institute of Science and Technology, JP), Yoshihiro Murata (Nara Institute of Science and Technology, JP), Naoki Shibata (Shiga University, JP), Keiichi Yamasato (Nara Institute of Science and Technology, JP), Minoru Ito (Nara Institute of Science and Technology, JP)

- A Middleware for Implementation and Evaluation of Application Layer Multicast Protocols in Real Environments
  Kazushi Ikeda, Thimme Baduge, Takanori Umedu, Hirozumi Yamaguchi, Teruo Igarashi (Osaka University, JP)
QoS in Wireless Mesh Networks: Challenges, Pitfalls, and Roadmap to its Realization

Parag S. Mogre, Matthias Hollick, Ralf Steinmetz
Multimedia Communications Lab (KOM)
Dept. of Electrical Engineering and Information Technology
Technische Universitaet Darmstadt
Merckstr. 25, 64283 Darmstadt, Germany
{pmogre, mhollick, rst}@kom.tu-darmstadt.de

ABSTRACT

Wireless Mesh Networks (WMNs) have the potential to lead to a disruptive change in the landscape of wireless communications. The vision to support self-engineered wireless network infrastructures that allow for an organic growth of the network is to be realized by means of self-organizing mechanisms for network configuration, control and optimization. Supporting a Quality of Service (QoS) to enable a rich portfolio of applications and scenarios is foreseen to be vital for the success of next generation WMNs. Today’s cutting edge standards supporting WMNs (e.g. IEEE 802.16’s mesh mode and IEEE 802.11s) are not perfectly equipped to cater to this task. While providing the necessary flexibility as well as sophisticated protocol mechanisms, these standards come with an inherent complexity and suffer from innate problems with respect to QoS provisioning. As a result, care needs to be taken when developing algorithms for supporting QoS on top of the standard’s mechanisms or when deploying such WMNs. We believe that a holistic approach is necessary to tackle the challenge for truly enabling QoS in WMNs. Failure to do so will result in inefficient performance and, in the worst case QoS violations. This paper reviews the critical aspects that need to be considered using the IEEE 802.16-2004 standard’s mesh mode as a case-study. In addition to the research challenges, we highlight pitfalls and give pointers to realize QoS in WMNs.

1. INTRODUCTION

In the recent years the deployment of WMNs has been looked upon as an upcoming and promising step towards the goal of ubiquitous broadband wireless access. WMNs are interesting not only in the context of small community networks and neighbourhood networks, but also in the area of enterprise-wide networks or wireless backbone networks that can be established in an ad hoc manner, e.g. in disaster recovery scenarios. QoS is a critical issue especially in the latter two scenarios. Mission-critical applications depend on the provision of adequate QoS support when deployed in the WMN. Network providers who look at WMNs as a cheap alternative to expand their existing wireless network infrastructure without incurring exorbitant deployment costs also look at WMNs as a viable alternative. In such networks, the providers wish to support the integrated services they already offer on their traditional wireless platforms. These applications such as voice and video over IP need to be provided with carrier-grade QoS support.

The current modus operandi towards QoS provisioning in the Internet, namely that of over-provisioning of bandwidth and other resources, is not applicable to WMNs. In particular, due to the broadcast nature of the wireless medium, wireless networks need to deal with the fundamental issue of interference and noise, which is not an issue in wired networks. In contrast to traditional single-hop cellular networks, multi-hop networks such as ad hoc networks and WMNs introduce additional challenging issues, which emphasize the problem of interference, simply due to the fact that a multi-hop route needs more transmissions as compared to a single-hop connection.

Thus, bandwidth is a precious resource in wireless networks in general. WMNs are usually considered to have low (or no) node mobility, where individual nodes in the WMN are usually not considered to be energy constrained. These favourable characteristics of WMNs open up avenues for cross-layer optimization which are not found in ad hoc networks or wireless sensor networks, where the high mobility or the restricted energy available at nodes restricts the use of sophisticated optimization strategies.

In this paper we critically analyze the mechanisms provided by state-of-the-art WMNs to support QoS. We look into the practical aspects of deploying algorithms for providing QoS within these networks. Fundamental issues that need to be considered when optimizing the performance of the mesh network are reviewed. Without loss of generality, to present our arguments in realistic settings, we base our discussion on the MeSH1 mode of the IEEE 802.16-2004 standard [11] for deploying a QoS-aware WMN.

The rest of the paper is organized as follows. Related work is introduced throughout the entire work. In Section 2 we discuss selected application scenarios that demonstrate the need for QoS in WMNs. In Section 3, we review the QoS-

1Throughout this document the notation “MeSH” refers to the optional mesh mode of the IEEE 802.16-2004 standard.
capabilities of the MeSH mode and look at the critical issues and trade-offs that arise when deploying QoS. We discuss important challenges as well as selected pitfalls in enabling QoS in WMNs in Section 4. There, we also sketch possible solutions to some pertinent questions. This is followed by a conclusion and pointers to further research in Section 5.

2. SELECTED APPLICATION SCENARIOS FOR WIRELESS MESH NETWORKS

Applications for WMNs can be manifold. We first introduce a (non-exhaustive) list of applications for WMNs. To structure the presentation, we distinguish between (1) applications for closed user-groups such as enterprise users or users in emergency response operations, (2) semi-closed user-groups served by network providers/operators, and (3) open groups of end-users as experienced in community networks or a wireless Internet:

- Carrier-grade voice and video conferencing, online gaming, video streaming.
- Community networks, neighbourhood services, Internet connection sharing, localized P2P applications.
- High-bandwidth data (file transfer), interactive data (e.g. web browsing).
- P2P applications: communications, distributed computing, distributed file backup, gaming.
- Wireless backbone for campus networks or emergency response networks.
- Wireless sensing, monitoring and control for critical infrastructures.

Mechanisms to support QoS in WMNs should be designed and deployed keeping in mind these applications. Next we discuss selected applications in the context of the aforementioned scenarios for the deployment of WMNs; we highlight the QoS requirements for those scenarios.

*Enterprise Perspective.* This category of WMNs is deployed by an enterprise or an organization to serve as a broadband wireless backbone providing backhaul services. Examples for such networks are wireless backbone networks for a campus area, or for an organization. Additionally, such a broadband wireless backbone WMN may be deployed in order to respond to disasters or emergencies. Here, the WMN provides the backbone for communication between the different rescue teams, which communicate using wireless hand-held devices. The WMN is then responsible for supporting QoS to enable several services and collaboration among the responders and the command and control center. It may be used for simple P2P communication, timely event notification, data transfer as well as to remotely control actuators such as robots that are deployed in hazardous zones. QoS support in such application scenarios is critical with most applications having strict delay bounds as well as bandwidth requirements.

*Operator/Provider Perspective.* WMNs are an interesting alternative for network providers who wish to expand the coverage of their existing network infrastructure incrementally, and without incurring exorbitant planning and deployment costs. Rural areas without network connectivity can be quickly provided with a broadband wireless network coverage using WMNs. Subscribers are increasingly demanding triple-play services over a wireless network and in addition seek support for subscriber mobility. WMNs deployed to meet the above goal need to feature mechanisms which allow the support of carrier-grade Voice over IP (VoIP), and other interactive and equally demanding applications such as video conferencing, video streaming and online multiplayer gaming. In addition, subscribers expect and require high-bandwidth connections to enable Peer-to-Peer (P2P) applications. Thus, the provider is interested in QoS mechanisms which enable the differentiation of multiple services in the WMN, while providing hard QoS guarantees for critical services.

*End-user Perspective.* Community wide WMNs are built usually using customer-operated equipment; they grow in an unplanned and organic manner. There may or may not be a central managing entity for the WMN. Individual nodes may join and leave the WMN at will. Community mesh networks are a cheaper alternative to traditional wired networks for the users. In areas lacking extensive network infrastructure this may be vital to provide Internet access to entire communities, despite the fact that only one or a few nodes have direct access to the Internet. Such networks may be used to support P2P data exchange among neighbours, for community-wide information services, neighbourhood patrol, etc. Most of the applications in this scenario do not have strict end-to-end delay bounds but profit when high bandwidth is available. QoS issues in such networks are therefore optimization of the capacity usage and supporting delay sensitive network applications such as conversational traffic. Additionally, QoS support in such networks has to consider security and trust issues. QoS support, thus, involves finding reliable and dependable routes from the source to destination. Also, because of the extensive use of P2P services and applications, QoS support needs to consider the challenges for P2P systems in traditional networks.

Other classifications of WMNs are possible. Next generation WMNs are expected to comprise heterogeneous devices with differing capabilities, and, thus, require support for multiple services with differing QoS requirements. In all the above scenarios, QoS support is crucial for the success of the applications. Currently, we witness a trend towards P2P communication and services. Hence, traditional centrally managed QoS provisioning is not sufficient in the long run, but distributed provisioning of QoS should be part of the research agenda.

Ref. [1] gives a comprehensive overview of WMNs and also highlights some important research challenges in deploying WMNs. There exists today a plethora of proprietary wireless mesh networking solutions. However, proprietary solutions tend to involve high costs and hence are limited to small scale deployments. Most of these WMNs use the basic IEEE 802.11 [10] standard which suffers from various shortcoming if used for WMNs (see [21]). In addition, the basic 802.11 standard [10] does not provide means to provide delay and bandwidth guarantees effectively. The current state-of-the-art standards for deploying WMNs [11, 9] enable explicit reservation of bandwidth for individual links in the WMN. As a result, if sufficient bandwidth is available


and is reserved on all links along the path, end-to-end QoS guarantees can be given. The IEEE 802.16 standard provides sophisticated mechanisms to support QoS provisioning and also foresees the support of different scheduling services catering to the demands of different application classes. The 802.11s proposal currently uses a handshake mechanism similar to the distributed scheduling mechanism outlined for the IEEE 802.16-2004 standard to support bandwidth reservation.

Standardized solutions are also expected to help further reduce the deployment costs for WMNs. In this paper we will look at the IEEE 802.16-2004 as a prototype state-of-the-art standard providing mechanisms to support QoS. Hence, we will present the challenges for deploying a QoS-aware WMN in the context of the 802.16 standard. We first provide a short review of the 802.16 standard’s QoS and bandwidth allocation mechanisms in the MeSH mode. This is followed by a discussion of the QoS challenges and a presentation of the roadmap to facing these challenges.

3. QOS IN THE IEEE 802.16 MESH MODE REVIEWED

QoS in the IEEE 802.16 MeSH mode is supported on packet-by-packet basis using parts of the mesh connection identifier (MeSH CID) present in each MAC Protocol Data Unit (PDU) to decide the per-hop handling for the packet. The fields enable the specification of parameters like the the priority, drop precedence and the type of the data. In addition the MeSH CID is used to specify if a packet needs to be retransmitted if lost. Thus, the mechanism is sufficient to deploy DiffServ [2] like per-hop forwarding behaviour in the mesh network. However, such a mechanism is useless if no means are available to guarantee availability of sufficient bandwidth for individual links in the WMN. To enable this, the standard uses Time Division Multiple Access/Time Division Duplex (TDMA/TDD) with spatial reuse to allocate bandwidth to individual links and to share the available bandwidth between different nodes.

Figure 1: Sample MeSH topology

The time axis is divided into frames where each frame is composed of a control subframe and a data subframe. The control subframe is used to transmit management messages required for network maintenance and bandwidth management. Data is transmitted only in the data subframe. To allow shared access to the medium in the control subframe, it is divided into a number of transmission opportunities. A data subframe is on the other hand split into a number of minislots (a minislot is the smallest unit of bandwidth allocation). The control messages are used to request and allocate bandwidth to individual links. The standard provides a range of mechanisms to allocate bandwidth as well as to manage the request and granting of bandwidth. We next briefly outline these mechanisms using the sample WMN topology shown in Fig. 1.

The 802.16 standard supports the following bandwidth allocation mechanisms in the MeSH mode, which are introduced below.

1. Coordinated centralized scheduling
2. Coordinated distributed scheduling
3. Uncoordinated distributed scheduling

Coordinated centralized scheduling. Here bandwidth allocations are centrally managed by the Mesh Base Station (MeSH BS). The MeSH BS specifies a scheduling tree (shown by the thicker arrows in Fig. 1) rooted at the MeSH BS. Nodes in the scheduling tree forward their bandwidth requests to the MeSH BS which in turn allocates bandwidth for the uplinks and the downlinks in the scheduling tree. Coordinated centralized scheduling cannot be used to allocate bandwidth for links not included in the scheduling tree (e.g. here link between nodes F, 1 and K, 2 is not included in the scheduling tree). Besides, the performance of coordinated centralized scheduling degrades with the growth of the scheduling tree (see Ref. [16] for a performance analysis considering real-time traffic constraints).

Coordinated distributed scheduling. Here nodes schedule (coordinate) their transmissions within their two-hop neighbourhood such that conflicting transmissions are not scheduled. The nodes use a three-way (request, grant, grant-confirmation) handshake protocol for reserving bandwidth for a link. To enable the computation of a conflict-free schedule each node locally maintains the status of individual minislots based on the information it has obtained from the neighbours about their schedules. Nodes obtain information about transmissions/receptions scheduled by their neighbours from the handshake messages. Thus, the correct reception of the transmitted handshake management messages is critical for the functioning of coordinated distributed scheduling. The IEEE 802.16 standard specifies a distributed mesh election algorithm for accessing the transmission opportunities in the control subframe such that when a node transmits no other node in its two-hop neighbourhood (at least) transmits simultaneously. This ensures that the neighbours of a node are able to correctly receive the transmitted control messages with a high probability. The coordinated distributed scheduling handshake messages are transmitted in the control subframe using the mesh election algorithm. Ref. [3] gives a performance analysis of the distributed scheduler in the MeSH mode.

Uncoordinated distributed scheduling. Here nodes use the same mechanisms as in coordinated distributed scheduling except for the mesh election algorithm. Handshake mes-
sages are not transmitted in the control subframe but in the data subframe. Transmission of messages takes place in minislots already reserved for the links in question (i.e. link from the data transmitting node to the data receiving node and vice-versa). Thus, the nodes do not have to wait to win a transmission opportunity using the mesh election algorithm. This type of scheduling is useful for the quick allocation of bandwidth. However, as nodes in the neighbourhood are not aware of the handshake (and hence the minislots allocated using the handshake) they may schedule conflicting transmissions.

**Figure 2: Bandwidth reservation (allocation) concept**

Fig. 2 visualizes the concept of bandwidth allocation in the MeSH mode. Using the above scheduling schemes a range of minislots can be allocated to individual links for a contiguous range of frames, where the number of contiguous frames is chosen from the set allowed by the reservation persistence shown in Fig. 2.

For further details about QoS support in the MeSH mode as well as details about the scheduling schemes readers are referred to [11, 15]. Ref. [15] gives a detailed overview of the above scheduling schemes in addition to further detail about the MeSH mode.

4. **QoS CHALLENGES, PITFALLS, AND POSSIBLE SOLUTIONS**

In this section, we present QoS solutions based on realistic assumptions, i.e. we use the requirements of the above scenarios and consider standardized WMN technologies as a framework for the QoS solutions. While a variety of academic research problems persist, we consider addressing real-life challenges (including implementation aspects) as the next step to make QoS-aware WMNs a reality. Our approach, thus, is to start with existing state-of-the-art standards and augment practically feasible extensions also ensuring compatibility. While it is a good idea to use a level of abstraction when designing algorithms and solutions, we recommended that the implementation aspects are kept in mind right from the start to lead to technologically sound solutions which can be realized in practice.

We maintain the above perspective when presenting the QoS challenges. We use the 802.16 MeSH mode as a prototype state-of-the-art WMN standard with built-in basic QoS support. Hence, without loss of generality, we outline the challenges, pitfalls, and a roadmap to solutions within the above context. In the following we present selected challenges and highlight the crucial aspects that need to be taken into account when designing solutions.

**Differentiation of Service, Interworking.** To enable various applications, the differentiation of service is crucial. The IEEE 802.16 standard identifies for the point-to-multipoint (PMP) mode of operation the following scheduling services: Unolicited Grant Service (UGS), Real-time Polling Service (rtPS), Non-real-time Polling Service (nrtPS) and Best Effort (BE) (for details about the scheduling services see [11]). The current MeSH mode specifications do not allow the realization of such scheduling services easily, however. Thus, a QoS architecture and mechanisms need to be designed within the framework of the MeSH mode to support sophisticated scheduling services. Moreover, interworking of the QoS mechanisms with higher layers such as IP has to be addressed. The incompatibility between the PMP and MeSH QoS specifications needs to be bridged to allow seamless interoperability. This is especially of interest for network providers who may initially deploy a PMP network and may wish to later adapt it to a mesh when the coverage area needs to be extended. Ref. [14] presents a QoS architecture to enable the deployment of scheduling services compatible with the PMP. In addition, sound mechanisms to service the packet transmission queues have to be deployed to ensure that QoS requirements of applications are met (see Ref. [20] for example).

**End-to-end QoS Provisioning.** The MeSH defines per-link, per-hop QoS mechanisms. However, for carrier-grade QoS support it is necessary to address an end-to-end regime within the mesh. In addition to the per-link mechanisms, QoS-aware routing mechanisms have to be considered (see [4, 5, 18] for QoS routing issues). Most of the literature, however, investigates the performance of QoS routing protocols solely for the contemporary, contention-based MAC protocols such as IEEE 802.11. For the novel reservation-based MAC technologies such as IEEE 802.16, the bandwidth reservation schemes cannot be decoupled from the QoS routing protocols, but play a significant role in the obtained routing performance. A cross-layer ([17, 19]) approach is needed to make effective use of the MAC layer mechanisms when provisioning end-to-end QoS. We believe that further work in this area is required to analyze the dependencies across layers and to find an optimal solution jointly across all protocol layers.

**Efficient and Adaptive Bandwidth Management.** When different traffic classes are supported (corresponding to the different scheduling services) the bandwidth reservation has to adapt to the needs of the applications. We need optimal schemes to schedule the long-lived and QoS-critical flows, while balancing the load in the network such that dynamic traffic demand can be fulfilled adequately. Static bandwidth reservation schemes which do not consider the class of traffic will lead to inefficient bandwidth usage leading to additional problems like unfair bandwidth distribution among nodes as well as wastage of reserved bandwidth resources. Ref. [14] proposes one such class-aware bandwidth reservation scheme. The presented scheme makes effective usage of the various distributed scheduling options provided by the MeSH mode to enable effective bandwidth reservation and scheduling services similar to the PMP mode. Similar suggestions to support QoS can also be found for the PMP mode (see [13]). In addition novel concepts such as networkcoding may be applied to WMNs to increase the traffic that can be supported by the WMN (see [12]). Further research in this area is needed.
Optimal Values for the Standard’s Parameters. In addition to the scheduling mechanisms, the MeSH mode involves several other protocol states and parameters, which need to be optimized. For example, the choice of parameters for the distributed mesh election algorithm highly influences the performance and delay of the three-way handshake used to set up per-link bandwidth using the coordinated distributed scheduling mechanisms [3]. Protocols for network entry and handover also need to be optimized. Non-optimal choice of parameters severely affects the time required for a new node to join the network, and in scenarios like an emergency-response will lead to an unacceptable delay for setting up the WMN. Ref. [8] investigates the performance of the network entry in the IEEE 802.16 standard and shows how critical the correct choice of values for the standard parameters is. Other mechanisms in the standard need to be critically analyzed and optimized. For operation in open and organically growing mesh networks, the self-adaptation of operational parameters needs further attention.

Security and Dependability Issues. The quest for security in wireless mesh networks is a challenging one. Especially in open and unplanned WMNs which grow organically, issues of dependability need to be addressed [7]. Self-stabilizing protocols in combination with trust and reputation management mechanisms for the individual nodes in the WMN play a critical role. Lack of security is a serious issue, and lays waste to all efforts expended in providing QoS. In closed mesh networks for critical infrastructures, guaranteeing QoS even under attack is of paramount interest. The IEEE 802.16 standard specifies a security sublayer which is responsible for enabling per-link encryption and security mechanisms. However, these mechanisms are not adequate to guarantee security from an end-to-end perspective.

Mobility and Physical Layer Issues. Features such as mobility support or elaborated PHY mechanisms such as concurrent usage of multiple wireless channels, directed antennas, etc. add to the complexity of the problem. E.g. the MeSH standard currently does not adequately support mobility. This again emphasizes the need for solutions that enable interworking and compatibility between standards that are feasible for a (static) MeSH backhaul network and technology to support user-mobility.

In addition to the above issues various further optimizations are possible (see [6] for example). We perceive that a holistic approach to QoS provisioning is needed for next-generation WMNs. Neglecting any of the above aspects when designing comprehensive QoS solutions will lead to severe problems. In addition to the above, features like self-configuration, self-healing and self-optimization of WMNs should be considered from the QoS perspective.

5. CONCLUSION

Wireless Mesh Networks are foreseen to lead to a disruptive change in wireless communications. However, to be successful, a set of criteria has to be fulfilled: a critical mass of subscribers/users need to be present, applications need to be adequately supported, secure and dependable operation has to be guaranteed. However, one crucial factor—possibly the tipping point—in making WMNs a success story is the support for QoS. In contrast to the wired Internet, bandwidth is scarce and over-provisioning cannot be applied to WMNs; thus, without adequate QoS mechanisms a lot of promising applications is likely to fail.

We propose the following roadmap for realization of QoS in WMNs. First, one has to derive the QoS requirements from application scenarios, and to analyze and state the assumptions that are induced by the employed wireless technologies/standards. The selection of an optimal combination of application and tool (i.e. state-of-the-art standard for WMNs) is the next step to reach the set goal. Second, we strongly believe that a holistic approach is needed to have enough flexibility to address the challenge of enabling QoS for WMNs. We opt for a cross-layer perspective, because optimization at one protocol layer needs to consider the trade-offs and influences at the other layers. Third, while designing the mechanisms, one should strive to keep the solution as simple and transparent as possible. This is particularly true, if we keep realistic implementation aspects in mind. The standard’s mechanisms need to be optimally harnessed, as well. Such an approach is vital for QoS to successfully accomplish the transition from theory to practice in WMNs.

6. REFERENCES


Enhancing Game-Server AI 
with Distributed Client Computation

John R. Douceur
Microsoft Research
Redmond, WA
johndo@microsoft.com

Jacob R. Lorch
Microsoft Research
Redmond, WA
lorch@microsoft.com

Frank Uyeda
Univ. of California
San Diego, CA
fuyeda@ucsd.edu

Randall C. Wood
Microsoft Game Studios
Redmond, WA
randallw@microsoft.com

Abstract
In the context of online role-playing games, we evaluate offloading AI computation from game servers to game clients. In this way, the aggregate resources of thousands of participating client machines can enhance game realism in a way that would be prohibitively expensive on a central server. Because offloading can add significant latency to a computation normally executing within a game server’s main loop, we introduce the mechanism of AI partitioning: splitting an AI into a high-frequency but computationally simple component on the server, and a low-frequency but computationally intensive component offloaded to a client. By designing the client-side component to be stateless and deterministic, this approach also facilitates rapid handoff, preemptive migration, and replication, which can address the problems of client failure and exploitation. To explore this approach, we develop an improved AI for tactical navigation, a challenging task to offload because it is highly sensitive to latency. Our improvement is based on calculating influence fields, partitioned into server-side and client-side components by means of a Taylor series approximation. Experiments on a Quake-based prototype demonstrate that this approach can substantially improve the AI’s abilities, even with server-client-server latencies up to one second.

1 Introduction
In Massively Multiplayer Online role-playing Games (MMOGs), the artificial intelligence (AI) that controls monsters and non-player characters is exceedingly poor. It is common for gamers to complain of monsters that are so stupid as to make the game unchallenging [22]. Despite popular belief, the fundamental problem is not that game developers cannot write better AI algorithms. Rather, the problem is that the servers that host MMOGs have insufficient computing power to support the computational demands of thousands of even moderately sophisticated, concurrently running AIs [19]. Adding more back-end server resources could solve the problem, but at a cost that would be prohibitive given MMOG operations economics.

MMOGs usually run on servers under control of a game operator, who bears the direct financial burden of computation and operations. Switching to a client-based infrastructure [9,13,17] would provide computational power that scales with the count of concurrently active players. However, this approach sacrifices the security and control that is generally considered necessary in persistent-world games. This is especially true in light of exploits that have extracted real money from MMOGs [4,15].

In this paper, we consider an approach to harnessing the aggregate computational power of client machines for improving game AI, for the purpose of making monsters behave in more interesting ways and thus improving the MMOG playing experience. Rather than suggesting a radical restructure of MMOG architecture [6], we propose supplementing server-based computation by offloading components of AI onto client machines.

This approach is not without challenges. It adds a substantial communication delay to code that normally executes within the game server’s main loop. It relocates critical functionality to clients that may fail or become disconnected. And, it makes sensitive computations more readily exploitable by unscrupulous players who hack their client software.

We focus specifically on the problem of latency, which we address with the mechanism of AI partitioning: splitting an AI into a server-side component that retains critical tight-loop control and a client-side component that computes tuning parameters for the server-side component. We do not specifically address the issues of client failure or exploitation; however, these problems are addressable by migration and replication. AI partitioning can enable migration and replication by making the client-side component stateless and deterministic.

As a demonstration of feasibility, we apply AI partitioning to the latency-sensitive problem of tactical navigation. We develop an improved navigation AI based on summed influence fields, and we offload the bulk of the computational effort as a 2D Taylor series approximation. Experiments demonstrate that the technique is effective even with latencies in excess of one second.

The next section presents our vision for improving the gameplay of MMOGs with sophisticated AI. Section 3 surveys issues that client offloading must deal with, including the issue of latency. Section 4 introduces AI partitioning, a mechanism for addressing the latency problem. Section 5 applies the AI partitioning technique to the problem of enhanced tactical navigation, which we evaluate in Section 6. We briefly survey related work in Section 7 and conclude in Section 8.
2 A vision of sophisticated AI

We envision MMOGs in which monsters and non-player characters display behavior that is complex and interesting, though well within the reaches of present AI technology. We imagine monsters that travel in packs across a wide range, that engage in useful or distracting activities rather than just milling around, that become aware of players by sight, sound, or smell, and that stalk their victims and pounce when unexpected. Sophisticated monsters will work together, attacking the same target and coordinating their efforts. They will assess a group of players collectively, deciding whether to attack based on an assessment of comparative strength.

By contrast, current AIs exhibit astonishingly simple behavior. When unaware of nearby players, a typical monster either waits in a delineated region or roams along a predetermined path. When a player comes within a defined distance, the monster launches a direct attack. When severely wounded, some monsters will fight to the death, whereas others will try to retreat via a simple path.

The problem of retreating is illustrative. A severely wounded monster should not run directly into its attackers, who may be roughly surrounding it. If some of the attackers are wielding melee weapons, it should try to stay out of their weapons’ range. It should try to get quickly away from attackers with range weapons, such as archers, who will likely be standing still. These influences collectively suggest a particular instantaneous direction the monster should head. However, the influences will change as players move and prepare attacks, so the optimal direction will rapidly vary as the monster retreats. The calculations involved are not highly complex, but they are computationally demanding, particularly for a server that is determining the appropriate next step for thousands of monsters concurrently.

3 Issues in client offloading

Sophisticated AI calculations can be offloaded to clients only if several issues are addressed, including the availability of client CPU capacity, communication latency between clients and the server, the possible failure of client machines, and the risk of client exploitation.

For clients to contribute substantially to computation of game AI, there must be enough spare CPU capacity on the clients that the local users’ gameplay is not disrupted by the additional computation load. The popular World of Warcraft MMOG requires a minimum CPU speed of 800 MHz and recommends a CPU speed of 1.5 GHz [3], yet 2/3 of desktop machines have CPUs with speeds in excess of 2 GHz [18,20]. This data suggests that there is ample spare CPU on game clients’ machines.

Offloading computation to a client may induce a substantial communication delay, as work that was previously performed in the server’s main loop is now distributed to clients, processed on those clients, and sent back to the server. Round-trip latency between access networks can reach 400 ms [23], and a 56K-dialup access network can add as much as 500 ms more [10]. Although some aspects of AI, such as high-level strategic planning, may tolerate latencies that approach one second, it is not a priori clear whether tactical-level AI can satisfactorily cope with the lag of such a network delay.

Client machines can also fail in various ways: They may crash or spontaneously reboot; network problems can cause intermittent disconnection; players may abruptly quit the game; or a competing client application might become active and leave little available CPU. The server can thus not afford to rely on any particular client to perform any given computation.

Furthermore, in the absence of a secured execution platform, AI code that runs on a client machine can be modified by the machine’s owner. The owner might weaken the AI to make monsters stupid and easy to kill, or strengthen the AI to make monsters smarter and readily able to kill competing players. The server cannot safely assume that clients will calculate results honestly.

4 AI partitioning

To deal with the problem of latency, we introduce the mechanism of AI partitioning. This technique can be made amenable to redundant computation, to address the problems of client failure and exploitation.

The key idea is to split the AI that controls each subject into two components: a server-side AI and a client-side AI. The server-side AI is fairly simple, highly tunable, and high-frequency in that it runs at the same rate as the server’s game loop. It performs any job that is intolerant of lag, such as targeting. The client-side AI is smart, complex, and potentially quite slow. Its function is to compute tuning parameters for the server-side AI, to support tasks such as long-term planning. The split between the two AIs isolates the server not only from the communication delay to the client but also from the computation delay of complex AI logic.

Periodically, the server sends a glimpse of the game state to the client, and the client responds with advice. A glimpse is a snapshot of limited scope, containing data of proximate relevance to the AI’s subject. This glimpse is input to the client-side AI computation. The output of the computation is advice for the server-side AI, typically in the form of parameters and coefficients. Because glimpses and advice consume bandwidth, it is important to keep them fairly small.

It is advantageous to design the server-side AI to be tolerant of stale advice. Communication delays, varying client CPU availability, and the possibility of client failure preclude any guarantee of promptly returned advice. Thus, the server-side AI should not require advice for any particular execution frame; rather, advice should be useful over a range of execution time. Punctuality may be beneficial, but it should not be critical. In the extreme case in which no advice is received for an extended period of time, the server-side AI should have a fallback mode in which it can operate independently of client advice.

There are several advantages to designing the client-side AI to be stateless. By this, we mean that each glimpse-advice computation is independent of prior computations, with no client-side state carried forward. If a client fails or becomes disconnected, and the server hands off the computation to another client, the new client can immediately pick up where the previous one left off. In addition, the server can temporally limit the effect of each client on the server-side AI by preemptively migrating the client-side AI: assigning successive computations for the same subject to different clients.
There are also advantages to designing the client-side AI to be deterministic, meaning that identical glimpses produce identical advice. To tolerate failures and exploits, the server can redundantly issue the same glimpse to multiple clients, effectively replicating the client-side AI. To deal with simple failures, the server can accept the first advice it receives. Alternatively, to deal with attempted client exploits, the server can wait for multiple replies and use plurality voting to determine the correct advice; however, this may increase latency as the server waits for replies from multiple clients. If the client-side AI computation needs to include randomness, the seed for the random-number generator should be selected by the server and sent with the glimpse to clients, thereby keeping the client-side AI deterministic.

5 Example – tactical navigation

To evaluate the feasibility of offloading AI to clients, we consider the problem of tactical navigation. This is a particularly challenging task because it is highly sensitive to latency. The intent of the example is not to demonstrate a particularly impressive AI; rather, it is to illustrate how AI calculations can be effectively partitioned in a way that tolerates the latency of remote computation.

5.1 Classic navigation

The conventional approach to game-AI navigation [19,21] is first to select a goal and then to move toward that goal via a series of predetermined waypoints. If the goal is an opponent, then when the opponent comes within a defined range, navigation switches to a mode of random selection among preprogrammed attack movements such as charging, feinting, and strafing.

The main benefit of this approach is computational efficiency, since the complex logic for selecting a new goal is performed sporadically rather than reevaluated on every frame, and detailed path calculations are performed offline prior to game execution.

However, this approach has at least two significant weaknesses. First, it only allows for one goal at a time. In contrast, humans can simultaneously weigh several goals and devise a path that optimizes over all of them. Second, this approach does not readily adapt to quickly changing circumstances, such as the virtual locations of teammates and opponents. Consequently, this approach cannot execute interesting and intelligent movement patterns, such as the complex retreating behavior described in Section 2.

5.2 Improvement: aggregate influence fields

We address the two weaknesses noted above with a more flexible approach to tactical game navigation. Rather than navigating toward a single selected goal, the approach is to calculate a vector field that characterizes the collective influence of all entities in the vicinity, and then to move in the direction indicated by the field. This field optimizes over both the explicit primary goal and implicit secondary goals, and it can be readily recalculated as entities move.

From the subject’s perspective, each other entity exudes an attractive or repulsive radial influence field with a magnitude of $|f| = W |d|^m$

where $d$ is distance from the subject, $m$ is a decay factor, and $W$ is a weight. Most entities are attractive, such as the primary goal, targeted opponents, weapons, ammunition, and health packs. Some entities may be repulsive, such as a powerful opponent who is currently attacking the subject.

The aggregate influence field is the sum of the fields from nearby entities. For a subject at point $p$ in virtual space, the aggregate field $f$ from a set of $N$ entities can be calculated as

$$f(p) = \sum_{k=1}^{N} W(k) \| p_k - p \|^{m(k)-1} (p_k - p)$$

where the weight $W$ and decay factor $m$ are functions that vary per entity. In general, the weight and decay functions are affected by the state of the subject; for example, as the subject’s health decreases, it becomes more repulsed by attacking opponents and more attracted by health packs.

We can reformulate the above expression for $f$ as

$$f(x, y) = \sum_{i=4}^{N} W(k) \left( (x_k - x)^2 + (y_k - y)^2 \right)^{-m(k)/2} \left( (x_k - x)i + (y_k - y)j \right)$$

where $i$ and $j$ respectively represent unit vectors in the X and Y dimensions. By simplifying the latter formulation, the cost of calculating the aggregate field is five additions, six multiplications, and one exponentiation per entity in the subject’s vicinity.

Fig. 1a illustrates a sample aggregate influence field. The subject is repelled by the attacking enemy, attracted to the other enemy, and more attracted to the health pack.

5.3 Offloading: Taylor-approximate fields

The cost calculating the aggregate influence field is proportional to the count of entities in the area. To offload the bulk of this effort to a client, we use AI partitioning, as described in Section 4. Specifically, in the server-side AI, we replace the calculation of the actual influence field $f$ with the calculation of a second-order two-dimensional Taylor series approximation:

$$f(\hat{x} + \Delta x, \hat{y} + \Delta y) \approx \left( A_0 + A_1 \Delta x + A_2 \Delta y + A_3 \Delta x^2 + A_4 \Delta y^2 + A_5 \Delta x \Delta y \right) i + \left( B_0 + B_1 \Delta x + B_2 \Delta y + B_3 \Delta x^2 + B_4 \Delta y^2 + B_5 \Delta x \Delta y \right) j$$

The cost of this computation is merely 12 additions and 13 multiplications, irrespective of the count of entities in the vicinity. The Taylor-series coefficients, $A_0$ to $A_5$ and $B_0$ to $B_5$, are computed by the client-side AI, based on a glimpse of the game state provided by the server when the subject is at point $(\hat{x}, \hat{y})$ in virtual space.
Fig. 1b illustrates a 2D Taylor series approximation of the aggregate influence field in Fig. 1a. Within the region highlighted by the circle, the approximation closely follows the actual influence field.

The edges of Fig. 1 show that the approximation can be wildly wrong, as the Taylor-series accuracy diminishes with distance from the point \((\hat{x}, \hat{y})\). Therefore, as the subject moves away from this point over time, the advice returned by the client-side AI becomes less valuable. In addition, even if the subject stays in place, the positions of the other entities will change, rendering the advice stale. We quantify this effect in the next section.

6 Evaluation

To determine how well AI partitioning can work in practice, we built a prototype of our Taylor-series-based tactical navigation. Because we want our experiments to evaluate partitioned AI performance even at high latencies, we do not provide a server-side fallback mode. We also do not implement replication; however, the client-side AI is both stateless and deterministic, so replication would be fairly straightforward to add.

6.1 Prototype implementation

Due to our unfamiliarity with available MMOG code bases [7,8], we evaluate the above mechanism in an open-source first-person-shooter (FPS) game, specifically Quake III. Although FPS games are quite different from MMOGs in many respects, the basic game loop and combat AI logic are very similar [1,19].

We built a prototype implementation of partitioned AI inside Quake III’s bot code. We employ Quake III’s standard AI [21] for every aspect of bot control except the direction of motion, which we determine by a Taylor-approximate influence field. Notably, we did not modify the target-selection logic or shooting accuracy.

Because our focus is on the AI-offloading technique, rather than on bot tactics, we did not perform extensive experimentation for optimal weights and decay factors in the field equations. Instead, we thought of a few basic strategies for tactical bot behavior. We then tinkered with parameter settings and parametric functions until we found a set that seemed to work reasonably well. The main considerations are that higher weights result in a greater influence, and higher decay factors increase the localization of the influence. Ultimately, we settled on four strategies of increasing sophistication:

**seekplayer:** The goal, identified by existing Quake III bot logic, is strongly attractive even at a distance \((W = 60, m = 1)\); other bots are attractive when they are nearby \((W = 20, m = 2)\); and items, such as weapons, ammunition, and health packs, are neutral.

**seekall:** Like seekplayer, except that nearby items are also attractive \((W = 20, m = 2.5)\) if considered useful. An item is designated useful if acquiring it will change the bot’s state: a weapon of a type not currently possessed, ammunition if the bot has less than the allowable limit, and a health pack if the bot has less than perfect health.

**selfaware:** Like seekall, except the state of the bot can cause some useful items to be designated as critical. This applies to weapons and ammunition when the bot has no ammunition left, and to health packs when the bot is below 20% health. Critical items are extra attractive \((W = 40, m = 2)\).

**avoidattacker:** Like selfaware, except that when the bot’s health is below 50%, it is repulsed by the last enemy to have damaged it recently. The repulsion is determined by whether the attacker wields a range weapon, in which case it is moderately repulsive when nearby \((W = -10, m = 2)\), or a melee weapon, which makes it more repulsive but only when very close \((W = -30, m = 3)\).
6.2 Experiments

Our virtual test environment is a flat space with no fixed obstacles, intended to resemble an open outdoor area as can be found in an MMOG. Since Quake environments must be bounded, we use the Quake map editor to build the largest allowable map, and we locate all items of interest in rough proximity to the center, to make the bots disinclined to go near the edges of the world. We also severely limit the availability of ammunition, to produce a relatively even balance of melee and ranged combat.

We evaluate our implementation by staging a series of six-bot, all-against-all “deathmatch” tournaments. Each tournament consists of nine ten-minute games, each involving three enhanced bots and three standard bots. By default, all bots are set to median skill (level 3 in the range from 1 to 5). To simulate wide-area network latency, we add a tunable delay to the client-server communication path. For each tournament, we measure the mean score difference between the enhanced bots and the standard bots. We vary the round-trip latency, the navigation strategy of the enhanced bots, and the skill level of the standard bots.

6.3 Results

Fig. 2 plots the difference between the mean scores of enhanced bots and the mean scores of standard bots, as a function of round-trip latency, for the four strategies enumerated above. It also plots 95% confidence intervals for these score differences. Even with latencies up to one second, our enhanced bots outperform the standard bots. We stress that this improvement has nothing to do with targeting or shooting accuracy, which we did not change. We replaced only the navigation logic.

The ability to tolerate a round-trip latency of one second suggests that the partitioning technique would be effective even in the high-latency environment described in Section 3. As the latency increases beyond this point, the standard bots begin to outperform our enhanced bots, as the advice from the client-side AI becomes increasingly stale. This suggests that a threshold of one second is a reasonable cutoff point for invoking a fallback navigation mode, as mentioned in Section 4. An obvious choice for fallback code is the standard bot navigation logic, which requires no advice from the client.

The confidence intervals for the four strategies overlap each other, implying that the four strategies are not significantly different in their effectiveness, or at least that any differences are too minor for our small set of experiments to demonstrate. Perhaps the ability to optimize over multiple goals, which is present in all four strategies, is the dominant factor responsible for the improved performance of field-based bots over standard bots.

We are somewhat surprised that sophisticated decision-making, as performed by the selfaware and avoidattacker strategies, has little effect on the game outcome, relative to our simpler enhancement strategies. However, this may be due to a quirk of FPS games that is not representative of MMOGs, namely that dying can be beneficial. A respawned bot regains full health and a partly loaded weapon, which is often an improvement over the bot’s state prior to being killed. Thus, a strategy designed to avoid dying may not improve game score.

Fig. 3 plots the score difference of level-3 enhanced selfaware bots versus standard bots of skill levels 3, 4, and 5. As already shown in Fig. 2, enhanced bots outperform standard bots of the same level even with latencies up to one second. In addition, at lower latencies, enhanced bots outperform standard bots of higher skill level, despite the fact that these higher-level bots have better target-selection logic and greater shooting accuracy. This is even more noteworthy considering that our replacement of the navigation logic had the side effect of removing classic combat maneuvers such as feinting and strafing, which the standard bots continued to employ.

Overall, our experiments show that an enhanced AI, partitioned into server and client components, can improve on an AI’s abilities, even with high round-trip latency.

7 Related work

Several prior researchers have investigated offloading workload from game servers to clients. Kubas et al. [11] forms clients into a multicast tree that disseminates server updates. FreeMMG [6] is a hybrid model, mainly peer-to-peer but with a server to help failed clients recover. Others have proposed completely replacing servers with peer-to-peer systems [9,13,17] running on clients.
In a different context, researchers have explored client-donated CPU cycles for distributed computing tasks, such as SETI@home and the other BOINC projects [2], and Folding@home and Genome@home [14].

The use of vector fields for directional guidance originated in the AI community, where it was applied to actual physical robots [5,12]. Mamei et al. [16] employed vector fields for strategically coordinating bots in Quake III; this is a sophisticated AI task to which our offloading technique could perhaps be applied directly.

### 8 Conclusions and future work

In this paper, we propose enhancing the AI of game servers by offloading computation to clients. To address the problem of latency, we partition each computation into a critical tight-loop server-side AI and an advice-giving client-side AI. As an exemplar, we design an enhanced AI for tactical navigation based on influence fields, and we partition it using Taylor series approximation. Prototype experiments show substantial improvement in AI abilities, even with round-trip latencies up to one second.

A further step with our prototype is to replicate the client-side AI and test its ability to deal with client failure. Another step is to add a local fallback mode to the server-side AI and investigate the transition between advised and unadvised AI behavior. A minor but practically important improvement is to execute the client-side AI on a low-priority thread, to ensure that it does not disturb the gameplay of the user on the client machine. Moreover, we would like to implement partitioned AI in an MMOG and conduct a user study to see whether it improves the game as we expect.

One aspect of client exploitation we did not consider is information leakage, in which clients inspect glimpses from the server to learn details of game state they should not be allowed to observe. We would like to investigate anonymization and obfuscation techniques to limit client visibility into offloaded computations.

### 9 Acknowledgements

The authors thank Mike Calligaro for insightful early discussions of ideas in this paper, Jeff Pang for his assistance with the Quake III code base, and the anonymous NOSSDAV reviewers for their very helpful suggestions in improving the paper.

### References


RACS: A Referee Anti-Cheat Scheme for P2P Gaming

Steven Daniel Webb  Sieteng Soh  William Lau
{steven.webb@postgrad,soh@cs,lauhow@cs}.curtin.edu.au
Department of Computing
Curtin University of Technology
Perth, Western Australia

ABSTRACT
Peer-to-peer (P2P) architectures provide better scalability than Client/Server (C/S) for Massively Multiplayer Online Games (MMOG); however, they increase the possibility of cheating. Existing P2P cheat solutions only prevent protocol level cheats, ignoring two prevalent forms of cheating: information exposure (IE) and invalid commands (IC). This paper proposes the Referee Anti Cheat Scheme (RACS), a hybrid between P2P and C/S. As in P2P, RACS allows peers to exchange updates directly, improving its scalability. However, similar to the server in C/S, the referee in RACS has authority over the game state, providing cheat resistance equal to that in C/S. This paper describes how RACS prevents cheating – including IE and IC. Our simulation and analysis show that the average bandwidth and delay in RACS is lower than that in P2P and C/S. This paper also includes a case study of integrating RACS with a commercial network game architecture.

General Terms

Keywords
Cheating, client/server, MMOG, peer-to-peer, referee, security.

1. INTRODUCTION
Massively Multiplayer Online Games (MMOG) differ from traditional network games as they present a single universe in which thousands of players participate simultaneously [9]. Most MMOG use a Client/Server (C/S) architecture, in which the server is the game authority whose tasks include: T1 - receiving player updates, T2 - simulating game play, T3 - validating and resolving conflicts in the simulation, T4 - disseminating updates to clients, T5 - storing the current game state, T6 - storing the offline-player’s avatar state, and T7 - authenticating players, downloading their avatar state, and billing. As the server is a trusted authority, addressing cheating, consistency, conflict resolution, and persistency issues is simplified. On the other hand, redirecting updates through the server (T1 to T4) adds game delay (response time), and consumes server bandwidth and processing power. While more servers can be provisioned, the required resources and financial costs grow rapidly with respect to the number of players, limiting C/S scalability [6,13].

Many Peer-to-Peer (P2P) architectures have been proposed to increase the scalability of MMOG [7,9], as each peer contributes its resources to perform tasks T1 to T5. However, P2P increases the possibility of cheating because it decentralises the game state to client machines.

Cheating is a major concern in MMOG [18] as it degrades the experience of the majority of players who are honest [12]. This is catastrophic for games using subscription models to generate revenue [5]. C/S provides strong protection against cheating, as the server has authoritative control over T2 to T5. Even so, some forms of cheating cannot be detected or prevented [17]. Converting to P2P moves the responsibility of tasks T1 to T5 to the peers, making cheat detection/prevention more difficult.

Several P2P protocols [2-4,6] have been proposed to solve protocol-level cheats. However, these protocols fail to address the information exposure (IE) and invalid command (IC) cheats (prevalent in MMOG [10,14]), and introduce new forms of cheating (e.g., the inconsistency cheat) not possible in C/S. In addition, the solutions require costly distributed validation-algorithms that increase game delay and bandwidth, which is economically undesirable since bandwidth is an expensive recurring cost [12]. Furthermore, the protocols in [3,6] introduce a new cheat, which we call the undo cheat (described in Section 2.2).

Several hybrid C/S and P2P systems [8,13] have been proposed to increase the scalability of C/S without reducing its security. Peer-to-Peer with Central Arbitrator (PP-CA) [13] has lower game delay and server outgoing-bandwidth since it allows peers to directly exchange updates; the CA receives all updates to resolve conflicts. However, PP-CA does not address cheating, and creates a new cheat (blind opponent (BO), discussed in Section 2.2). Furthermore, it does not use Area of Interest (AoI) filtering to minimise updates, a crucial component for scalable architectures.
In this paper we propose the Referee Anti-Cheat Scheme (RACS) that extends PP-CA. RACS uses AoI filtering, and solves the undo, BO, IE and IC cheats. RACS requires lower delay and bandwidth than either C/S or P2P cheat solutions, while providing security equivalent to C/S.

The layout of the paper is as follows. Section 2 describes various cheats and their solutions. In Section 3 we present the details of RACS. Section 4 provides an analytical and simulation study of RACS. Section 5 concludes the paper.

2. BACKGROUND

2.1 Cheats and Solutions

We consider a cheater who can eavesdrop, replay, modify, delay, insert, and destroy messages sent/received by his host. He may also modify any program or software on his system. Note, “he” should be read as “he or she” throughout this paper. We exclude general security issues such as authentication, denial of service, etc.

Table 1 classifies cheats into four levels: game, application, protocol, and infrastructure; some cheats fall into multiple levels (e.g., IE). The table extends that in [6] with the undo, BO, IE [10], IC [14], and proxy/reflex enhancers (PRE) [14] cheats. We consider PRE, such as aiming proxies, in the infrastructure level as they are deployed in the network between the client and the server [14]. Section 2.2 details the undo, IE, IC, and BO cheats; see [6] for the details of other cheats. Table 1 also shows the different cheat solutions. Note that the solutions are not mutually exclusive, e.g., using C/S with PunkBuster (PB).

<table>
<thead>
<tr>
<th>Cheat</th>
<th>C/S</th>
<th>PB/VAC2</th>
<th>AS</th>
<th>NEO/SEA</th>
<th>RACS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Game Level</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bug</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Application Level</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IE, IC</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Bots/reflex enhancers</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Protocol Level</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Suppressed update, Timestamp</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Fixed delay, Inconsistency</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collusion</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Replay, Spoofing</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Undo</td>
<td>NA</td>
<td>●</td>
<td>NA</td>
<td>NA</td>
<td>●</td>
</tr>
<tr>
<td>BO</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Infrastructure Level</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IE</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>PRE</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

Current P2P cheat solutions [2-4,6] divide time into rounds, each of which comprises two consecutive steps: (i) commit-to-an-update by transmitting an encrypted message, and (ii) reveal-the-update by sending the message key. To prevent cheating, all players must commit an update before any player reveals their key.

Lockstep [2] is too slow for many genres of games as it has a worst-case round length of 3d, where d is the delay between the two slowest players. Furthermore, it is vulnerable to inconsistency, replay, and spoofing cheats [6]. Asynchronous Synchronisation (AS) [2] increases responsiveness by only requiring players with overlapping Aos to work in lockstep. The round length in AS is reduced to between 2d and 3d; here d is the delay of the two slowest players with overlapping Aos. Unfortunately, AS does not prevent cheaters/griefers [8] from increasing d. Sliding Pipeline (SP) [4] is another variation of Lockstep that increases the transmission rate by pipelining updates; however, its worst-case delay remains at 3d.

New Event Ordering (NEO) [6] limits the round length of every group of players with overlapping Aos to 2d. NEO only considers an update valid if the majority of the group receives it within d; late updates are discarded. Each player then transmits the update’s key in the second half of the round. However, NEO is vulnerable to inconsistency, timestamp, replay, and spoofing cheats [3]. Secure Event Agreement (SEA) [3] modifies NEO’s cryptography to address these cheats, while maintaining its delay bound. Unfortunately, both approaches [3,6] suffer from the undo cheat (discussed in Section 2.2).

2.2 The IE, undo, IC, and BO Cheats

Information Exposure (IE): The goal of IE is to obtain secret information to which the cheater is not entitled, thus gaining an unfair advantage in selecting the optimal action. We have included IE in the application and infrastructure levels since information can be exposed at both levels, subject to how the cheat is performed. At the application level the game client or data files can be modified to reveal secret information, while at the infrastructure level, IE is achieved by modifying either (i) the graphics drivers to render the world differently [18] (e.g., drawing walls transparently), or (ii) the network infrastructure to allow another host to sniff the network traffic [14]. Reference [10] proposes On-Demand Loading to address IE, at the expense of additional processing.

Undo: Let $P_i$ denote a player with a unique identification $i$, and $M_i$ and $K_i$ represent a message and its key from $P_i$ respectively. Without loss of generality, the undo cheat is illustrated in Figure 1 involving only two players: an honest $P_h$ and a cheating $P_C$. Both players send their encrypted game moves ($M_{ih}$ and $M_{ic}$) in the commit phase. Then, $P_h$ sends key $K_{ih}$ in the reveal phase. $P_C$ cheats by delaying $K_{ic}$ until $K_{ih}$ is received, and $M_{ih}$ is revealed. If $P_C$ decides that his committed $M_C$ is poor against $M_{ih}$, $P_C$ will purposely drop $K_{ic}$ (dashed line) undoing his $M_C$. Even worse, in [3,6], $d$ is determined by the majority of players, which
allows colluding cheaters to increase the round length \(d\) to gain time for evaluating their opponent’s moves.

**Invalid Command (IC):** In IC, the client application or data files are modified to issue commands that originally could not be generated [14]. IC is trivially solvable in C/S since the server validates all commands. However, preventing IC in P2P is difficult because all peers (including cheaters) must agree on valid commands.

![Figure 1. Sequence of messages in the undo cheat.](image)

**Blind Opponent (BO):** A cheater in PP-CA may purposely drop updates to his peers (but not to the CA), effectively *blinding* them about his actions. Section 4.1 describes our RACS solution.

### 3. REFEREE ANTI-CHEAT SCHEME

#### 3.1 Concept and Protocol

RACS comprises three entities: an authentication server, a set of players \(\{P_i\} \; i\) is the unique identifier (ID) of each player, and a referee \(R\). The server is used to store offline-player’s avatar state (T6), authenticate joining \(P_i\), download \(P_i\)’s avatar state to his host and \(R\), and billing (T7). The server assigns a unique ID to each player. Each player receives updates (T1), simulates game play (T2), and sends updates to his peers and the referee (T4).

The referee \(R\) is a process running on a trusted host that has authority over the game state. Note that distributed referees will be addressed in future work. The referee performs Tasks T3 and T5 to prevent cheating and maintain the game’s consistency. For these tasks, it receives and simulates all updates (T1 and T2). The referee performs T4 if peers are unable to communicate directly (in the event of message loss or cheating, discussed in Section 4.1).

The referee divides game time into rounds of length \(d_{\text{MAX}}\); the developer sets \(d_{\text{MAX}}\) such that the game is playable. Note that \(R\) can decrease (increase) \(d\) to lower delay (reduce its outgoing bandwidth); the algorithm to compute the optimal value for \(d\), and the frequency of using it are application dependent, and therefore we do not address these issues in this paper. In general, \(d\) should be adapted to accommodate the clients’ Quality-of-Service (QoS) characteristics. We suggest setting \(d\) to the maximum client delay less than \(d_{\text{MAX}}\). However, we do not recommend changing \(d\) when high priority events are occurring, as this may induce temporal disruptions.

For each round \(r\) every \(P_i\) generates a pair \(U_i=(r, I)\), to be included in his messages transmitted to \(R\) and other peers. Here, \(I\) is the information containing \(P_i\)’s actions (e.g., move, attack, etc.) and/or information about connections with his peers (e.g., informing \(R\) about disconnecting from an opponent). The referee initialises the round number \(r=1\). Each copy of \(r\) (kept in \(R\) and each \(P_i\)) is independently incremented for every elapsed \(d\). One can use NTP [6] for synchronising \(d\).

As shown in Figure 2, RACS considers three different message formats: (i) peer to peer message – MPR, (ii) peer to referee message – MPR, (U, S, T), and (iii) referee to peer message – MRP, each of which is signed by the sender (i.e., \(P_i\) or \(R\)). It is obvious that MPP (MPR) is the smallest (largest). Note that MPP does not transmit secret information \(S_i\) (e.g., health and items), which is conceptually similar to On-Demand Loading [10]. Instead, \(S_i\) is only included in MPR to the referee. In addition, he includes a set \(T_i=[\{j, H(U), D(MPP)\}]\) so that the referee can detect inconsistency between each hash of update \(U_{ij}\), \(H(U)\), that \(P_i\) received from each of his opponents \(P_j\) (in the previous round) with that received by \(R\). For each unmatched \(H(U)\), \(R\) requests \(P_i\) to forward the MPP that he received to verify the cheat using the non-repudiation quality of digital signatures; this step prevents cheaters incriminating opponents by sending incorrect hashes to the referee. The referee uses the transmission delay of all MPP, \(D(MPP)\), to adjust \(d\). Receiving MPR, (U, S, T), \(R\) forwards MRP to \(P_i\)’s peers if the players are in PRP mode; otherwise (i.e., in PP mode), \(R\) simulates the game and only sends MRP to relevant players when inconsistency is detected. Note that PP and PRP modes are discussed in Section 3.2.

![Figure 2. RACS communication models](image)

**Figure 2. RACS communication models**

The recipient of each message validates its authenticity using the public key of the sender. A late message (not received within its round) is considered for a future round assuming no newer messages have been received; otherwise it is discarded. Thus RACS is more tolerable to slow players and network delay than [3,6] which discards late messages. We assume the use of a public key infrastructure for authentication and non-repudiation [6].

#### 3.2 Communication Models

As shown in Figure 2, the communication between any \(P_A\) and \(P_B\) that are mutually aware (within each others’ AoI) can be through the referee \(R\) (Peer-Referee-Peer: PRP mode), or direct (Peer-Peer: PP mode). In PRP each player sends MPR and receives MRP messages to/from \(R\). This mode provides security equal to that in C/S. In contrast, peers in PP exchange their messages (MPP) directly, which
reversion requirement for cases (i) and (ii), if the missing messages, or (iii) he
receives less than p percent of P’s last s ≥ 1 messages, or (iii) he does not receive P’s update for more than w ≥ 0 consecutive rounds. Reversion requirement (i) provides AoI filtering to reduce bandwidth; only players that include P in their AoI will be updated. Requirement (ii) prevents a cheater repeatedly sending one message and then dropping w consecutive messages, while requirement (iii) ensures that losses are not clustered, which would have a large impact on the game-play experience. For either case, P sends an MPP (MPR) to P (R), that includes I notifying them of the reversion. Then, R only forwards P’s moves to P if P is within P’s AoI. Note that RACS is cheat-proof when w = 0 or p = 100%. The optimal values for w, p, and s should (i) minimise PP to PRP reversions, and (ii) minimise the number of messages that may be dropped.

Each leaving P (in PRP or PP) sends MPR (with I = QUIT), which makes R upload P’s avatar state to the server, which in turn, sends an acknowledgement (ACK) to both P and R. Receiving the ACK, P disconnects from R, and R notifies all affected players.

Every P_i in PP mode sends his MPP (MPR) to all affected peers (R) for each elapsed d. Thus, for every round, R (each player) expects a message from each player (all other players). However, due to communication failures or cheating, a message may not arrive. Assume P sends a message to P_A. We consider three cases for missing messages: (i) neither receives, (ii) only R receives, and (iii) only P_B receives a message. In RACS, P and R dead-reckon the avatar of each peer whose message is not received. R’s state is authoritative, and it notifies affected players about inconsistencies caused by dead-reckoning. In case (i), only P_A may be disadvantaged, as P_B and R have matching state. However, in case (ii), P_B might be slightly disadvantaged if his game state is incorrect. Finally, case (iii) disadvantages both P_A and P_B since R’s dead-reckoning may make their states incorrect. Note that for cases (i) and (ii), if the missing message violates reversion requirements (ii) or (iii), P will revert to PRP.

Every P_i in PRP mode sends MPR (U_i,S_i,T_i) to R for each elapsed d. In the following round, the referee sends MRP(R,U_i,i) to P_i’s peers. The referee (peer) dead-reckons P_i’s avatar for each missing MPR (MPR); any inconsistency is resolved using the R’s authoritative state.

4. RACS PERFORMANCE EVALUATION

4.1 Security

PRP mode provides security equivalent to that in C/S, since both models use a trusted entity to simulate the game and forward updates. Obviously cheat solutions in PRP are similar to those in C/S and, thus, are not discussed in this paper. In the following, we explain how RACS in PP mode addresses various cheats. Throughout our discussion, we assume a referee R, a cheating P_C, and P_C’s opponent P_H.

Bugs: RACS assumes that bugs will be fixed by software patches (released by the publisher), as does C/S.

IE: P_C does not receive any secret information S_H, which is only included in MPR; thus IE is prevented.

Bots/reflex enhancers: As in C/S [5], one may combine RACS with PB or VAC2 to detect bots/reflex enhancers.

IC: R validates all player commands to prevent IC. Further, the authentication server stores all offline players’ avatar state to prevent command tampering.

Suppressed update: Missing messages are dead-reckoned/interpolated by P_H and R. Since R’s state is authoritative, P_C gains no advantage from suppressing his updates.

Fixed delay: Applying fixed delay may make P_C violate reversion requirements (ii) or (iii), which, in turn, will revert P_H from PP to PRP (with respect to P_C). This will punish P_C because his delay (with respect to P_H) will be two hops, while that of P_H (with respect to the other PP peers) will be one hop. If P_C’s message arrives within the round (not late), RACS does not consider it cheating. Since late updates are indistinguishable from cheating, the solution may penalize honest but slow players; nevertheless, it is better than [3,6] which prohibit slow peers from playing.

Inconsistency: R detects inconsistency by comparing the hashes of all the messages received by P_H in the previous round with those received by R from P_C (see Section 3.1).

Timestamp: Late updates are considered for the following round (Section 3.1), and thus the cheat is prevented.

Collusion: RACS does not address the collusion cheat, nor do existing P2P and C/S solutions [17].

Replay attack: A message is invalid if another message with larger r has been received (see Section 3.1); hence, r is a nonce to detect replay of old messages.

Spoofing cheat: RACS solves this cheat by authenticating the signature of the message.

Undo: RACS does not use the commit/reveal steps as in [3,6], and thus the undo cheat is impossible.
BO: This cheat is equivalent to missing message case (ii) (Section 3.2). If the cheat results in $P_C$ violating reversion requirement (ii) or (iii), $P_M$ will revert to PRP, which solves the cheat. Otherwise $P_C$ gains an insignificant advantage.

4.2 Analytical Evaluation

Our analysis assumes one update (size $L_a$ bytes) for every $d=T_w$ seconds. We consider a game with $N$ players, and every player is aware of at most $M$ opponents. The worst (best) case for RACS is when no (all) peers use PP. A peer sends $L_a/T_a$ ($L_a/T_a$)*$M$ bytes in the worst (best) case. For both cases, the inbound traffic to the referee and each peer is ($L_a$)*$N$ and ($L_a$)*$M$/$T_a$ bytes, respectively. Thus, the inbound bandwidth requirement of RACS is equivalent to that of C/S, and may potentially be its bottleneck. The best case outbound traffic from the referee is negligible since MRP is only sent to resolve inconsistencies. In the worst case, the referee sends ($L_a$)*$N$ bytes, equivalent to the server in C/S. The AoI filtering, used in RACS, greatly reduces the outgoing bandwidth of the referee and peers compared to the server and peers in PP-CA, while maintaining the same delay. Thus, RACS is more scalable than PP-CA.

RACS is superior to NEO/SEA [3,6] in several ways. First, RACS requires lower bandwidth. Every message in SEA is comparable in size with each MPR, and is sent to all peers in the same region. In contrast, peers in RACS send MPR, only to the referee, and use MPP (smaller than MPR) between peers. Thus, the peer bandwidth in RACS is lower than in SEA. Second, RACS determines dynamic $d$ more accurately and faster with lower bandwidth. Here, NEO/SEA use a distributed algorithm [6] to calculate $d$, in contrast to our centralized method that requires lower bandwidth cost. Further, directly setting $d$ to the maximum client delay computes it more accurately and faster. Finally, in contrast to RACS, NEO/SEA requires complex group selection algorithms to prevent cheaters from colluding against a minority of honest players.

4.3 Simulations

We used the Network Game Simulator (NGS) [16] (netgamesim.sourceforge.net) to show the impact of cheaters on RACS delay and bandwidth requirements. We considered a game with a world of size 5000 by 5000 units, and a referee that handles 5000 players with an AoI radius of 50 units. Avatar movement is controlled by the random-way-point mobility model with a velocity of two units per second and a wait time of 0. We simulated 1000 seconds for a lossy network with no late messages. Each player generated a message every $d=50$ms. Further, after reverting to PRP, a pair of peers will not attempt PP mode for at least 60 seconds. We set $w=0$ (thus, $p$ and $s$ are irrelevant) to show the worst case bandwidth and delay costs of RACS in preventing cheats; equivalently, $p=100\%$ might be used. Here, the cheaters do not send MPP messages, as in BO and suppressed update. These cheats have the greatest impact on the average delay and R’s outgoing bandwidth because they require the referee to forward updates ($U_i$ in MRP) to other peers. We varied the percentage $c$ of cheaters, randomly selected, from 0% to 100%.

Figure 3 shows the referee’s average and maximum outgoing bandwidth per second (left Y axis) and the average delay (right Y axis) with an increasing percentage of cheaters. The average bandwidth (delay) was calculated by dividing the total bandwidth used (delay of all messages) by the length of the simulation (number of messages), whereas the maximum bandwidth was the peak bandwidth consumption per second measured in the simulation. As expected, the figure shows that the best case (worst case) occurs when $c=0\%$ (c=100%) as all updates are exchanged directly (routed through the referee) using PP (PRP) mode. Figure 3 shows that RACS scales well, even in the presence of cheaters, as honest players continue to exchange updates. Note that the average bandwidth and delay of RACS never exceeds those of C/S ($c=100\%$).

![Figure 3. RACS with increasing cheaters.](Image 368x410 to 507x496)

The following simulation illustrates how a developer sets optimal $w$, $s$, and $p$ for a lossy network. We consider the Source Engine (SE) in HalfLife (HL) [15] to show how the values are determined. Note that HL requires very low delay [4], and thus this illustration is applicable to all genres of games, including MMOG. In the SE an update is generated every 50 milliseconds. As most client’s delay exceeds this, messages must be pipelined. Note that RACS adopts a similar pipelining approach to that in [6] to avoid any security issues. The SE does not render received updates for 100ms, and uses interpolation to smooth player transitions. In the event of two consecutive message losses, the SE client dead-reckons for up to 250ms; therefore, the client halts after seven consecutive losses.

From the described specifications, we set $d=50$ms and $w=6$ ((100ms+250ms)/50ms - 1 = 6); hence, a peer reverts to PRP after seven consecutive losses. We believe that losing 2*w=12 messages per 10 seconds will give a cheater an insignificant advantage. Thus, $s=200$ (10 seconds/50ms), and $p=94\%$ (11/12/200 * 100%). The simulation uses MPP message loss rates from 0% to 50%. We assume the communication to/from the referee is well provisioned, and therefore the loss rate for MPR and MRP is insignificant. Also, all messages arrive on time or not at all. Since the
effects of modifying $w$ and $p$ can only be observed when players have repeated interactions, we simulated 12 players, each with an AoI radius of 200, in a world of size 100 by 100 so that all players are constantly mutually aware. We simulated the SE with 0 ($c=0\%$), 3 ($c=25\%$), and 6 ($c=50\%$) cheaters. We use all remaining parameters from the previous simulation.

Figure 4 shows the average bandwidth and delay of the simulation. The figure also includes a worst-case baseline, i.e., $w=0$, $p=100\%$, $c=0\%$. With no cheaters, it is obvious that increasing $w$ and reducing $p$ greatly reduce the referee’s outgoing bandwidth and average game delay (highest vs. lowest plots in the figure).

The figure shows that RACS is highly tolerant to loss. Irrespective of the number of cheaters, loss rates below 20% do not impact the outgoing bandwidth and the average delay; beyond 20%, RACS performance degrades rapidly. We believe that the critical point is caused by the values of $p$ and $w$, and therefore further investigation is required for tuning the parameters. However, the results show that increasing numbers of cheaters has a greater impact on performance than loss rate; as more peers revert to PRP mode. Nevertheless, the upper bound of RACS delay is two hops (as in C/S), below SEA’s three-hop bound. In addition, as discussed in Section 4.2, its overall bandwidth never exceeds that in C/S and SEA.

**Figure 4.** RACS with increasing message loss and cheaters.

5. CONCLUSION

We have extended the cheat classification in [6] and proposed RACS, which has the following benefits: (i) it provides security equal to C/S, while reduces delay and the server/referee’s outgoing bandwidth; (ii) it is more effective and efficient than existing cheat solutions [2-4,6], as it is secure against the IC, IE, undo and BO cheats with lower cost; (iii) it allows peers with poor connections to play using PRP, unlike [3,6], and (iv) its centralised algorithm calculates $d$ more accurately, faster and with lower bandwidth than the distributed algorithms in [3,6].

As with PP-CA, RACS reduces only the outgoing bandwidth; it does not address the scalability issues of incoming bandwidth and referee processing requirements. We are investigating the use of multiple referees in a server cluster (similar to Federated C/S) and/or in peers to reduce the referee’s incoming bandwidth and processing requirements; hence, improving RACS scalability. Distributing referees to peers increases scalability; however, this raises issues of referee trust, selection, load balancing, and synchronization. These issues require further investigation.

6. REFERENCES

[11] Li, J., & Kang, X. mSSL: Extending SSL to Support Data Sharing Among Collaborative Clients. in Proc. ACSAC’05, pp. 397-408.
Traffic Analysis Beyond This World: the Case of Second Life

Stênio Fernandes
Centro Federal de Educação Tecnológica de Alagoas
Maceió, AL, Brazil
stenio@gprt.ufpe.br

Rafael Antonello, Josilene Moreira,
Djamel Sadok
Universidade Federal de Pernambuco
Recife, PE, Brazil
jamel@gprt.ufpe.br

Carlos Kamienski
Universidade Federal do ABC
Santo André, SP, Brazil
cak@gprt.ufpe.br

ABSTRACT
Virtual Worlds (VW), such as Massive Multiplayer Online Social Games, have been gaining increasing attention in the last few years, mainly due to the new way users interact with them. However, little effort has been devoted to understand their traffic profile and the implications to the traffic management area. With the current growing rate of VWs’ usage, their traffic demand could eventually impose a significant burden on the operation of a typical Internet Service Provider (ISP) network. In this paper, we seek to understand the traffic behavior of an increasingly popular VW application, namely Second Life, from both the connection and network level perspectives. We also show results of a traffic analysis of a Second Life client, when an avatar performs different actions in the virtual world, at different places and under different network conditions. Our results show that Second Life makes intensive use of network resources (mostly bandwidth), since the capacity needed for having a full second life experience (listening to live music) may reach 700 Kbps.

Categories and Subject Descriptors
C.2.5 [Local and Wide-Area Networks]: Internet; H.4.3 [Information Systems Applications]: Communications Applications; K.8.0 [Personal Computing]: General - Games

General Terms
Measurement, Virtual Worlds

Keywords
MMORPG, Second Life

1. INTRODUCTION
A Virtual World (VW) is an interactive simulated environment accessed by multiple users, represented by avatars, through an online interface [1]. Virtual worlds (or digital worlds) provide new levels of socialization, where users can experience sensations and interact with others in a similar way to real life. New forms of expressions for human behavior, fun and amusement, and most importantly, they can conduct business offered by such environments. Furthermore, there is the advantage of preserving privacy and anonymity. Second Life (SL) [2] is a popular VW application that has been gaining a lot of attention from the news media since last year and is also one that keeps growing in the number in terms of financial dealings and users (also called residents).

Virtual worlds (like SL) offer their residents the opportunities for building new places, often comprised of terrain, buildings and objects where users seek services just like in real life. They differ from traditional online games, where players have a particular goal to achieve and are limited to interact with the preexistent environment. Second Life provides a scripting language (called LSL) that allows residents to add objects with additional features to the world.

These features make Virtual Worlds very exciting for designers, programmers and users, but raises concerns for transport and content providers who have to provide adequate quality of service levels to guaranty a good interaction experience. In addition to the traffic sent from the virtual world’s server to the clients, programmers may trigger the streaming and download of real-time of other types of traffic coming from different servers from all over the Internet cyber space. In Second Life, many places allow users to activate music streams, coming from servers maintained by independent owners and sites. Such capability has the potential to generate many different traffic profiles (e.g., voice, music, video, and data) from a single virtual world, causing considerable impact on a typical ISP network. Nonetheless, little effort has been devoted to understand Second Life’s traffic profile and the implications to the traffic management area.

In this paper, we take a first step to understanding the traffic behavior of Second Life, from both the connection and network level perspectives. We show results of a detailed traffic analysis of a Second Life client, while an avatar performs different actions in the virtual world (standing still, walking and flying), at different places (both popular and unpopular), using different access network technologies (ADSL and university LAN) and with or without external traffic sources (e.g., while listening to music or not). We collected information at different weekdays and hours, summarizing more than 100 hours of Second Life usage with different avatars. However, due to the space constraints, only the most significant findings are presented.
Our preliminary results show that Second Life makes intensive use of network resources (mostly bandwidth). The traffic bitrate arriving at the client with a full second life experience (flying with live music) reached sustained levels of about 500 Kbps and peaks of up to 700 Kbps. In some places, music accounted for about 200 Kbps (probably encoded at 192 Kbps), which comes to emphasize our argument that external data sources may generate unpredictable and potentially harmful traffic patterns.

The rest of the paper is structured as follows. Section 2 discusses related work. Section 3 presents the architecture of Second Life, from a connection and client-server level point of view. Section 4 exposes the methodology of our work and section 5 the results we obtained. Finally, section 6 draws some conclusions and presents topics for future work.

2. RELATED WORK

With the current growing rate of Massively Multiplayer Virtual World (MMIV), specifically of Massively Multiplayer Online Role Playing Game (MMORPG), such as Everquest and World of Warcraft, networking researchers have been struggling to understand their network requirements through the analysis of their traffic profiles [5][8][10][11]. An in-depth knowledge of their traffic behavior will certainly assist Internet Service Providers (ISP) to provision and better design their network infrastructure. In [8], Kim et al present traffic measurements of a popular MMORPG (Lineage II) and provide its traffic characterization from the server side point of view. In [9], Feng et al analyzed the network traffic of several popular online games with the focus on First-Person Shooting (FPS) game traffic, namely Counter-Strike. Claypool and Claypool [10] evaluated the effects of Internet latency on online games. They argued that their results are useful for game designers (e.g., by applying different latency compensation techniques), network designers (e.g., by creating infrastructures providing quality of service (QoS) for interactive applications) and game players (e.g., by providing knowledge for QoS purchases). We observe that most MMORPG and FPS described in the literature have low bitrate requirements, due to its intrinsic characteristics of sending frequent but small packets, mainly for synchronization issues. Although the aggregate traffic at the server could eventually impose a high burden for ISPs, from the client side perspective, downstream traffic are typically far below the capacities of typical broadband connections. Indeed, most of these environments allow dial-up users to have a fair playing experience.

However, a slightly different type of MMORPG, called Massively Multiplayer Online Social Game (MMOSG), such as Second Life (SL), has recently become hype and, to the best of our knowledge, its communication patterns are still unknown. Our work reveals that the bandwidth requirements for SL are far beyond from those for popular MMORPG and FPS games. As of April 2007, SL has more than 5.5 million registered users and an average of 30,000 simultaneous on-line users. With the recent widespread deployment of broadband access technologies, the increase of the number simultaneous online users is putting pressure on both SL servers and ISPs network infrastructure and demanding an in-depth traffic analysis of the SL’s network behavior.

3. SL ARCHITECTURE

This section presents the roles of different servers in the SL architecture, their interactions between themselves and the way they exchange data with clients.

3.1 Servers

SL is based on an asymmetric client-server architecture, where each server is dedicated to a particular task such as login, instant messaging handling, or region simulation. The SL world consists of a great deal of interconnected and uniquely-named simulators (sim). Each sim is a process responsible for processing a 256x256 meter region, and communicates only with its four nearest neighbors, thus avoiding the transactional scaling problem as the world becomes really large [5]. A grid is a graphical division of units representing the servers which run SL, wherein one server supports one sim (although multiple sims may be, and actually are, executed on the same physical server). The user is known as viewer. As the viewer moves through the world it is handed off from one sim to another [3]. Figure 1 depicts the SL architecture, including the current known servers.

![SL Architecture Diagram](http://www.secondlife.com/whatis/economy_stats.php)
computations on objects and land and transmit these data to the viewers. Physics simulation is handled by the Havok physics library [4]. Simulators communicate with their neighbors via UDP connections.

**Other servers:** Central Database (CDB) contains a list of who owns what, e.g. used for billing. Agent Database keeps track of the mapping between metadata and item id (UUID), which is a globally unique identifier (128-bit number). Inventory database contains information about user’s assets, which is a data resource such as an image, sound, script and object. Assets can be downloaded to the viewer or uploaded into the central asset store. Search Database is a replica of the Central DB used for search. The Map server renders the overall map with OpenGL. The RPC server behaves as an API for developers to manipulate Second Life without using the viewer. It translates XML-RPC server into in-world requests and communicates with the space and CDB servers.

### 3.2 Authentication Flow

Figure 1 depicts the 8 steps required to establish a user connection to Second Life, which are:

1. The viewer sends a secure message checksum request to the server (port 12036).
2. The viewer sends a XML-RPC function call, over https to login. The server can suggest an optional update.
3. The Login server queries the database server for authentication credentials.
4. The Login server decides which sim to send the viewer to. Then, requests session start.
5. The sim sends a reply to the login server with the verification that user is allowed on the region.
6. The login server sends a reply to the viewer with agent id, session id, secure session id, sim IP, sim port, global location and some inventory information.
7. The viewer sends user id and session id to the sim, as a handshake. The simulators address range is 64.128.0.0 to 64.129.255.255, and the port range is 13001-13050.
8. The viewer sends presence information to the User server, so that instant messages among group members may be exchanged. The User server checks group rights and sim session information.

### 4. DATA COLLECTION METHODOLOGY

We visited different places at different hours and days, summarizing more than 100 hours of experiments using SL. Data for this paper were collected from January 24 to 29 (2007). The methodology for our experiments and the configuration are explained in the next paragraphs.

**Scenarios:** After visiting a large number of different places in SL, we realized that the number of avatars and objects in a place makes a significant difference in the traffic. Therefore, we considered two places using the criterion of popularity. We used “Goddess of Love” and “Menglin II” for representing a popular and an unpopular place, respectively. In order to estimate the level of attractiveness of a place (popular or unpopular), we utilized the traffic index found in the SL client. According to SL documentation, such metric is a measure of the proportion of the in-world time that other avatars have chosen to spend in a specific place.

**Network Connections:** We repeated the same experiments for two types of network connections, namely a 100 Mbps link (called UFPE) from our University to the Brazilian Research Network, and a residential ADSL connection of 600 Kbps (called ADSL).

**Actions and external sources:** Each experiment was repeated for 3 different actions of the avatar, standing still, walking and flying. We also took into account external traffic sources (e.g., media streaming servers). Therefore, we repeated the experiments with and without music.

**Metrics:** We evaluated a number of metrics, although not all of them are presented in this paper, due to space constraints. We use the tcpdump protocol analyzer for capturing packets while using the SL viewer. The metrics considered were throughput, packet size (including TCP, UDP and IP headers), packet inter-arrival time, traffic volume (UDP/TCP).

For each experiment, we considered only a period of 10 minutes and the metrics were computed for each 10 second interval. In other words, for each 10 minute experiment we have 60 samples of the metric being evaluated. Most results present the average of the 60 samples and the 99% confidence intervals we also computed, although the pictures do not show them.

### 5. PERFORMANCE EVALUATION AND RESULTS

#### 5.1 Throughput

Figure 2 and Figure 3 present the time series for the throughput of UFPE and ADSL, with the audio stream turned on or off, and only for a given popular place. Observing the throughput collected for those scenarios, one may notice that different network connections do not significantly affect the amount of traffic generated by the application. In other words, even in a scenario with an external music source, a 600 Kbps ADSL downlink is enough for providing a good experience for the users in most cases. Figure 2 shows that the average throughput is around 300 Kbps for the ADSL connection, whereas within the university network (UFPE) it is around 400 Kbps with peaks of up to 700 Kbps.

![Figure 2 – Throughput (Flying, with music)](image)

On the other hand, when the client has deactivated the audio stream, Figure 3 shows an even lower difference between ADSL and UFPE. Throughput for the former is around 180 Kbps whereas it reached 210 Kbps for the latter. From these data, it is clear that SL makes an intensive use of network connections.
resources and for enjoying a full experience, a user needs a broadband connection of at least 500 Kbps, if an external audio stream is desired. Such requirements are beyond most MMORPG and FPS games.

From now on, due to space constraints we will only consider the scenarios for the residential network connection (ADSL) and without an external data source (e.g., a media streaming).

Figure 3 – Throughput (Flying, no music)

Figure 4 – Throughput (average)

Figure 4 shows the average bandwidth usage for 3 actions (standing still, walking and flying) and 2 places (popular and unpopular). Regardless of the action of the avatar, the popular place generates at least 2.5 times more traffic than the unpopular place, revealing that the bandwidth usage has a strong relationship with the particular place in SL where the avatar is.

There is also a clear correlation between the motion pattern of the avatar and the generated throughput. In Figure 5, an empirical cumulative distribution function (ECDF) shows that for an unpopular place, when the avatar is standing still, up to 97% of the time the throughput is below 20 Kbps, whereas with some form of movement it is between 60 Kbps and 110 Kbps in 87% of the time.

As far as SL places are concerned, a comparison between Figure 5 and Figure 6 points out that the throughput is much higher for the popular place (up to 400 Kbps) than for the unpopular place (up to 150 Kbps). Even when the avatar is standing still in the popular place, 93% of the time, the throughput is between 75 and 350 Kbps, which is much higher than that for the unpopular place.

5.2 Packet size

Figure 7 presents the mean packet size for both upstream (client-to-server) and downstream (server-to-client) traffic. In addition, Figure 8 and Figure 9 show the distribution of the packet sizes for the same scenario. In this case, the mean packet size has a similar profile as that observed by Chen [11]: in general, server-to-client packets are bigger than the client-to-server ones.

Please note that the average packet sizes generated by both the server and the client at the unpopular place are similar. This is possibly due to the fact that in a unpopular (with few people and objects), the amount of information the server sends to the client is smaller due to the lack of details of the surrounding environment.

Figure 5 – Throughput distribution (unpopular place)

Figure 6 – Throughput distribution (popular place)

Figure 7 – Packet size (bytes)
5.3 Traffic Volume

We found out that both TCP and UDP transport protocols are used by the application. The communication between SL clients and simulator servers use UDP, whereas the external audio streaming sources use TCP. Figure 10 shows that TCP packets are only exchanged in scenarios with music. The total volume of TCP bytes collected in our experiments is nearly the same (i.e., about 5 MB) for the 3 different actions: standing still, walking and flying. As expected, it is similar for both popular and unpopular places, which suggest the external servers may have used equivalent codecs for the audio stream. Although external traffic sources may generate unpredictable traffic patterns, as they are not controlled by SL, during our experiments they generated a steady sending rate (around 192Kbps).

5.4 Packet Interarrival Times

The average packet inter-arrival times (ADSL) are presented in Figure 11. It may be observed that the interarrival time is shorter for more complex environments (i.e. popular places) and when the avatar is moving faster. Obviously, this happens because either information concerning location and objects (including obstacles) sent to the viewer need to be updated more frequently or more information is sent in each update. In a simple environment the number of objects and avatars is smaller and consequently fewer packets have to be sent per unit of time. Figure 12 and Figure 13 corroborate with our findings, since they scrutinize the packet size and interarrival times correlation.

Figure 12 shows that when the avatar is standing still in an unpopular place, the client sends small packets (50 to 150 bytes) to the server, whereas the server updates the client with slightly larger packets (100 to 250 bytes), mostly below 200ms of interarrival time. On the other hand, a closer look at Figure 13 reveals that the profile for the client-to-server communications remains almost the same, whereas the server definitely increase the size of the packet payload, which we believe to contain additional information about other avatars and details of the surrounding objects. Although we do not show here, the motion pattern of the avatar also has a strong influence on both inter-arrival times and packet sizes from server to client. The client sends every movement to the simulator that in turn sends back updated information of objects, textures, other avatars, 3D viewpoint and so on.
6. Discussion
By filling a niche that is of interest of the Internet research community, this paper reveals that the traffic profile from the SL client’s point of view, can generate diverse network patterns. This is mainly due to the fact that within SL, users are allowed to customize the virtual world, by building an assortment of objects with distinct network requirements. Each object may have unique characteristics that require additional bandwidth in order to be rendered properly by the SL client. In addition, users can attach external traffic sources within their land boundaries, such as live streaming radio. Traditional MMPORG games differ from SL given that most of them have fixed scenarios and objects, and are not customizable. Such characteristics allow a more steady and predictable traffic behavior between client and servers.

By the analysis of some selected metrics, collected at the client over a long period of time, our work provides a first picture of how developers, designers and researchers in both networking and virtual environments fields can improve the performance of their systems or networks. For instance, by the knowledge of traffic patterns of the SL clients, a local ISP could monitor and forecast the aggregate bandwidth requirement to plan a link capacity upgrade. Alternatively, ISP could apply traffic shaping and policing mechanisms.

Although we did not focus on the comparison of SL traffic patterns with other 3D online games, it is clear that SL requires more stringent network parameters. In fact, most games of different types (e.g., FPS, RPG, and Social) share common behavior with SL, such as position update messages between client and servers. However, in most games the 3D scenarios are built within the client software, which alleviate the exchange of information about the surrounding objects near the avatar. In this case, clients and servers only need to exchange information about avatars’ position. For instance, a study of a MMORPG game [11] concluded that the average bandwidth required per client is small (e.g., around 7 kbps), whereas for Counter Strike it is about 40 Kbps [9]. SL requires bandwidth around 400 Kbps for the popular places and up to 150 kbps for unpopular places.

7. Conclusions
Second Life (SL) is a virtual world that has been increasingly gained the attention from the news media and Internet community since the last year. This paper takes a first step on understanding the SL architecture and profiling the traffic generated by its servers and clients. Our results show that SL makes intensive use of network resources, and for enjoying a full experience, a user needs a broadband connection of at least 500 Kbps (with external audio stream). Without a live audio stream, the average server-to-client throughput is about 200 Kbps for a popular place and below 100 Kbps for an unpopular place. This is within the possibilities of most broadband residential users and currently does not represent a threat for ISPs network planning. However, since SL permits developers to attach different external traffic sources, some SL places may generate higher and unpredictable traffic patterns in the future. Furthermore, the increase of the number simultaneous online users is a potential concern for network management.

As future work, we intend to derive traffic models for SL and comparing it with another similar virtual world. We envisage presenting a breakdown of the SL traffic by showing how many percent of traffic comes from chat messages, avatar position updates, transmission of 3D objects. Since SL client went open source and its messages are not encrypted, it opens up a number of possibilities for profiling SL traffic within SL client source code.

8. References
PAT: Peer-Assisted Transcoding for Overlay Streaming to Heterogeneous Devices

Dongyu Liu¹, Eric Setton², Bo Shen², and Songqing Chen¹
¹Dept. of Computer Science
George Mason University
Fairfax, VA 22030
{dliu1, sqchen}@cs.gmu.edu
²Mobile and Media Systems Lab
Hewlett-Packard Laboratory
Palo Alto, CA 94304
{eric.setton, bo.shen}@hp.com

ABSTRACT
With the increasing deployment of Internet P2P/overlay streaming systems, more and more clients use mobile devices, such as smart phones, PDAs, to access these Internet streaming services. Compared to wired desktops, mobile devices normally have smaller screen size, less color depth, and lower bandwidth and thus cannot correctly and effectively render and display the data streamed to desktops.

To address this problem, in this paper, we propose PAT (Peer-Assisted Transcoding) to enable effective online transcoding in P2P/overlay streaming. PAT has the following unique features. First, it leverages active peer cooperation without demanding infrastructure support such as a transcoding server. Second, as online transcoding is computationally intensive while the diverse devices used by participating nodes may have limited computing power and related resources (e.g., battery, bandwidth), an additional overlay, called metadata overlay, is constructed to instantly share the intermediate transcoding result of a transcoding procedure with other transcoding nodes to minimize the total computing overhead in the system. The experimental results collected within a realistically simulated testbed show that by consuming 6% extra bandwidth, PAT could save up to 58% CPU cycles for online transcoding.

Categories and Subject Descriptors
H.1 [Information Systems]: Miscellaneous

General Terms
Algorithms, Experimentation

Keywords
P2P/Overlay Streaming, Meta-Transcoding, Heterogeneity

1. INTRODUCTION

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

NOSSDAV ’07 Urbana, Illinois USA
Copyright 2007 ACM 978-1-59593-746-9/06/07 ...$5.00.

In recent years, the amount of Internet media traffic has increased exponentially. According to ComScore [1], from October 2005 to March 2006, the Internet video traffic doubled every 3 to 4 months, and the number of users watching videos online increased by 18% monthly. To efficiently deliver Internet media objects, a number of different P2P/overlay streaming systems have been proposed [6, 10, 12, 14, 16, 17, 18]. In practice, PPLive [2] regularly provides access to more than 400 TV channels daily and has been used by hundreds of thousands of users.

Recent technology advances in wireless and 3G have made Internet access pervasive. Many users today use PDAs or smart phones to access the Internet instead of traditional desktop computers. According to Telephia [3], more than 34.6 million mobile subscribers accessed the Internet via wireless devices in June 2006 and these 34.6 million subscribers account for 17% of the total 206 million US mobile phone subscribers.

While the usage of all kinds of mobile devices is increasing on the Internet, delivering streaming media to these devices in overlay streaming systems is challenging, since these mobile devices often differ from desktops in the screen size, the color depth, and the available downloading/uploading bandwidth. Therefore, the streaming data intended to a desktop cannot be directly rendered and displayed on a PDA or a smart phone.

Although some existing studies have considered that participating nodes may have different capabilities in uploading bandwidth in an overlay streaming system (e.g., [18]), which is not an uncommon situation for most residential broadband users if ADSL or cable is used for Internet connections, they have not considered the heterogeneity in multiple dimensions when various mobile devices join the system. Some existing solutions based on scalable/layered coding [7, 13] or multiple description coding [16] may not be effective given the fact that most devices do not support these codecs. On the other hand, in the context of P2P/overlay streaming, precoding requires either streaming of compound objects containing all versions or a separate overlay for each version, both of which imply waste of bandwidth.

In this paper, we propose PAT (Peer-Assisted Transcoding) to facilitate P2P/overlay streaming in heterogeneous environments. The PAT scheme relies on node cooperation without demanding infrastructure support. Media adaptation is carried on by real-time online transcoding at the peers. We show that this approach is effective in satisfying diverse demands from heterogeneous participating nodes in a P2P system.
Although the transcoding solution is more flexible in adapting to heterogeneous environments, one key challenge remains that such a solution is computationally intensive. Some research has been conducted to study online transcoding [4, 5, 8]. Most such schemes treat the transcoding procedure as a black box with the final transcoded result as the only caching candidate. In our previous study [11], we have introduced the idea of meta-caching. Meta-caching identifies intermediate transcoding steps from which certain intermediate results (called metadata) can be cached and reused later so that a fully transcoded object version can be produced from the metadata with a relatively smaller amount of CPU cycles. Meta-caching demonstrates an effective trade-off between the storage and CPU load. Such metadata-assisted transcoding is called meta-transcoding.

This meta-transcoding approach is also leveraged in PAT. In addition, the sharing of the metadata is facilitated by the construction of an additional overlay, called metadata overlay, which is in parallel with the overlay used for data streaming. This approach instantly shares the intermediate result of a transcoding procedure with other transcoding nodes, aiming to minimize the total computing overhead in the system.

With the assistance of meta-transcoding, PAT effectively improves streaming quality and reduces overall CPU load. Through extensive real-data-parameterized simulations, we show that the client-perceived streaming quality in PAT is significantly improved, and with a small amount of additional bandwidth (6%), PAT can significantly reduce the CPU load (up to 58%) for online transcoding.

To the best of our knowledge, this work is the first to combine transcoding and P2P/overlay streaming. We believe our proposed scheme is generic and it can be used to design both tree and mesh based transcoding assisted overlay streaming systems. In this paper, for illustrative purposes, the tree-based overlay streaming system is used as a base to introduce the scheme.

The remainder of the paper is organized as follows. Section 2 discusses related work. We describe the system in section 3. A performance evaluation obtained with a simulated network is presented in section 4. We make conclusion remarks in section 5.

2. RELATED WORK

Research on P2P/overlay streaming systems has attracted considerable attention [2, 6, 16]. Padmanabhan et al. [16] use multiple distribution trees and multiple description coding (MDC) to provide redundancy to the system and enhance robustness of the content distribution. Splitstream [6] is a high-bandwidth content distribution system where content is divided into $k$ strips in order to distribute the forwarding load among all the participants. P2P/overlay streaming using data-driven unstructured P2P networks has also been studied. For example, Chainsaw [17] provides unstructured solutions to high-bandwidth data dissemination systems. A number of studies have also investigated efficient topologies for P2P live streaming. For example, Small et al. [18] theoretically investigate the scaling law of P2P live multimedia streaming and their results can have practical implications when constructing tree topologies. A scalable architecture is proposed for congestion controlled multicast real-time communication by using self organized transcoder in [9]. Dagster [15] proposes a novel incentive scheme to encourage nodes to contribute more bandwidth to the whole system. Nodes in Dagster are assumed to be able to conduct transcoding during broadcasting. However, it is not clear how a peer could efficiently conduct transcoding at runtime.

3. PEER-ASSISTED TRANSCODING (PAT) DESIGN

In this section, we first briefly introduce the basics of meta-transcoding and metadata matrix. Then we present the design of Peer-Assisted Transcoding (PAT), which consists of two overlays. One is the base overlay, which is used for streaming data transmission. The other is the metadata overlay, which is used to instantly share metadata, the intermediate results of a transcoding procedure, to efficiently minimize the total overhead for online transcoding by trading a small amount of bandwidth for a large saving on CPU cycles in the system.

3.1 Meta-transcoding

In [11], we have proposed meta-caching to effectively balance the resources used for online transcoding between the CPU cycles and the limited storage space in a transcoding proxy or server. The motivation of meta-caching is as follows. Many existing schemes that aim to optimize transcoding treat the transcoding procedure itself as a black box, focusing on the optimization of caching transcoded object versions based on the prediction of future requests. However, a typical transcoding procedure involves multiple steps, among which some intermediate results, also called metadata, may be saved to avoid re-computing the same result again for a later transcoding request. If the saved metadata only consumes a small storage size while it takes a significant computing load to generate, it is apparently an effective trade-off between the storage and CPU cycles. For example, for bit rate reduction transcoding, the requantization scale factor can be stored rather than being re-computed each time data is transcoded. In this case, meta-transcoding requires significantly less CPU cycles to produce the new transcoded object.

In this paper, we combine this approach with P2P/overlay streaming, where a media stream is spread to a large population of clients, utilizing the forwarding capacity of the peers. In addition, a node can also instantly share some intermediate transcoding results with other nodes performing same transcoding. This approach reduces the CPU overhead in the system. Clearly, additional bandwidth is required for metadata sharing. In the following context, we propose a protocol that can reduce computing load with small bandwidth overhead.

3.2 Meta-data Matrix

We assume there is a single root (the source of the stream). In addition to providing the original streaming data, the root also maintains a small matrix. The size of this matrix is determined by the number of possible stream qualities. For example, if the video to be streamed has different qualities from 1 to $n$, then the matrix is a $n \times n$ matrix. Each element in the matrix indicates availability of the metadata corresponding to transcoding from version $i$ to $j$. By definition, the metadata matrix, denoted as $m_{Matrix}$, is an upper triangle matrix, since it is rare in practice to transcode a
lower quality version to a higher quality version. Figure 1 shows an example of an mMatrix.

\[
\begin{bmatrix}
0 & T_{0,2} & \text{null} & T_{0,3} & \cdots & T_{0,j} \\
0 & 0 & T_{1,3} & \text{null} & \cdots & T_{1,j} \\
0 & 0 & 0 & T_{2,4} & \cdots & T_{2,j} \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & 0 & \cdots & T_{j-1,j} \\
0 & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & 0 & \cdots & 0 \\
\end{bmatrix}
\]

Figure 1: Example of an mMatrix.

When the corresponding metadata is not available, \( T_{i,j} \) in mMatrix is null. Otherwise, such metadata is available and \( T_{i,j} \) denotes the ID of the node which transcodes the stream from version \( i \) to version \( j \). We also call this node as the meta-head and there may be other transcoding nodes that receive metadata from a meta-head to perform meta-transcoding.

3.3 Base Overlay and Meta-data Overlay Construction

The construction of base overlay in PAT differs from traditional P2P/overlays, mainly because when a node wants to join the streaming service, its parent selection procedure is substantially different from traditional ones and also affects the construction of the metadata overlay. The node departure in PAT also affects both the base overlay and the metadata overlay.

3.3.1 Node Arrival

Assume the system starts with a source node and several bootstrap nodes providing the streaming service. When a node wants to join the streaming service, it first contacts the source node or a bootstrap node with its quality requirement. In this study, the quality requirement is represented by the desired frame rate and bit rate version of the content.

We map the quality requirement into a version index. The joining node wants to join the streaming service, it first contacts the desired frame rate and bit rate version of the content.

The joining node starts to probe nodes receiving version \( t \), where \( T_{t,j} \) is null in mMatrix. The joining node checks the mMatrix from the source node or a bootstrap node by examining the \( j^{th} \) column of mMatrix. The possible parent candidates are from \( T_{t,j} \) to \( T_{t-1,j} \) (a total of \( j-1 \) candidates). Based on the type of transcoding, the joining node starts to send the joining request to the meta-head node indicated by \( T_{t,j} \) (\( t \) is a variable and \( 0 < t < j \))

- where \( (j-t) \) is the smallest, if the transcoding is for frame rate or spatial resolution reduction. This policy is designed to minimize the computing load since it requires less computing load to transcode from a frame rate or spatial resolution version that is closer to the target frame rate or spatial resolution version.
- where \( (j-t) \) is the largest, if the transcoding is only for bit rate reduction. This policy is designed to minimize the quality degradation since the transcoding between versions that are further away (i.e., less number of intermediate versions) from each other in bit rate leads to less generation loss. On the other hand, the computing load for transcoding between different versions in bit rate is almost the same.

Then the candidate parent \( P \) decides whether to accept the joining node based on its bandwidth availability. As \( P \) needs to conduct meta transcoding, metadata subtree \( T_{t,j} \) is updated as follows: the meta-head of \( T_{t,j} \) looks for a parent for \( P \). If the meta-head of \( T_{t,j} \) itself has enough bandwidth, it serves as the parent of \( P \); otherwise it will check whether any of its current children has enough bandwidth to accept the new child \( P \) using a breadth-first search. If \( P \) can find a parent to join in the metadata overlay, the joining process finishes. If none of these nodes can accept \( P \), the joining node will try another non-null element in mMatrix in ascending (or descending) order of \( j-t \) value depending on the transcoding type, and repeat the joining process. If all the non-null elements in the \( j^{th} \) column of mMatrix have been explored without a parent node being identified, go to the next step.

- Step 2 – metadata unavailable: If a parent is not found in the previous step, the joining node must look for a parent with enough bandwidth and CPU resource for transcoding. To minimize the overall system transcoding overhead, the joining node starts to probe nodes receiving version \( t \), where \( T_{t,j} \) is null in mMatrix. The joining node probes the node with the smallest or largest \( j-t \) value depending on the type of transcoding as discussed in Step 1. If no suitable parent is found, the joining node probes the next node in ascending (or descending) order of \( j-t \) value until a parent node is found. The chosen parent will start a full transcoding process. The chosen parent can also share the metadata produced during the transcoding session by joining the metadata overlay as follows.

- After the parent (with its ID denoted as \( P \)) that is willing to do transcoding is selected and the joining node joins the base overlay, \( P \) also needs to join the metadata overlay. Sup-
pose node $P$ needs to do transcoding from version $k$ to version $j$. As $P$ needs to do full-transcoding, metadata subtree $T_{k,j}$ is updated as follows: node $P$ will become the meta-head of the $T_{k,j}$ subtree and the $T_{k,j}$ in mMatrix is updated.

If a parent node can not be found, go to the next step.

- **Step 3** If version $j$ is not the lowest quality version, go back to Step 1 request version $j+1$. Otherwise, the joining request is rejected.

During the joining process, the joining node may find several eligible parents and it can save the information about the nodes (e.g., $s=3$) as its backup parents. The reason to do this is to increase the system robustness: the node can quickly connect to its backup parent when its current parent leaves the system.

### 3.3.2 Node Departures

In P2P/overlay streaming, participating nodes may depart at any time. Upon node departure, the overlay must be re-adjusted to adapt to the change. Such adjustments are performed as follows.

- **Case 2.1:** If the departing node is a leaf node and is receiving version $j$, denoting the parent of the departing node as $P$, (1) if $P$ is not a transcoding node, remove the departing node from $P$’s children list and update its bandwidth information; (2) if $P$ is a transcoding node conducting transcoding from version $i$ to version $j$, $P$ needs to depart from the metadata subtree $T_{i,j}$ according to the following META-LEAVE algorithm. If $P$ is the meta-head in the metadata tree, it tries to find a new meta-head from its children with the largest available bandwidth. If a child $C$ is selected, $T_{c,j}$ in mMatrix is updated accordingly and the sibling nodes of node $C$ become the children of new meta-head $C$ and parent-children relationship of the related nodes is updated. If $P$ is the meta-head with no children in the metadata overlay, $T_{c,j}$ in mMatrix is set to null. Otherwise, if $P$ is not a meta-head in the metadata tree, the meta-head deletes node $P$ from the metadata subtree. If node $P$ has children, they will become the children of the parent of $P$ in the metadata subtree if the parent of $P$ has enough bandwidth. They will contact the meta-head to rejoin the metadata tree if the parent of $P$ does not have enough bandwidth. The corresponding children and parent fields are updated in these nodes.

- **Case 2.2:** If the departing node is neither a leaf node nor a transcoding node, the children of the departing node need to find a new parent. If the parent of the departing node has enough bandwidth, it becomes their new parent. Otherwise, the children of the departing node check their backup parents. If the backup parents cannot accept these nodes as children, the children of the departing node need to perform a rejoin process as described in Section 3.3.1.

- **Case 2.3:** If the departing node receiving version $i$ is not a leaf node and is a transcoding node, the adjustment must be done on both the data overlay and the metadata overlay. We assume the departing node $D$ is in the metadata tree $T_{i,j}$ and the children of node $D$ need version $j$. All children of $D$ in both the metadata tree and the data tree need to find new parents. In the metadata tree, node $D$ departs the meta-data tree according to the META-LEAVE algorithm in Case 2.1. After the metadata tree is updated, the first child of $D$ and its siblings in the data tree find a new parent as follows. In the updated metadata subtree $T_{i,j}$, if there are still some nodes in the metadata tree $T_{i,j}$ after the departure of node $D$, they can be selected based on their available bandwidth. If the metadata tree $T_{i,j}$ is null or no node is available, the peers need to rejoin the multicast session. The node has a chance to subscribe to a node that can directly provide version $j$ within the data tree.

### 4. PERFORMANCE EVALUATION

In this section, we first perform real transcoding and meta-transcoding in our implemented transcoder for both bit rate reduction and frame rate reduction. With parameters obtained from these experiments, we evaluate the performance of PAT based on NS2.

#### 4.1 Experimental Parameter Capturing and Setup

In our implemented transcoder, we conduct both bit rate transcoding and frame rate transcoding. A 1000-second video with a data rate of 500 kb/s and 30 frame/second (denoted as f/s) is used as the original object for each transcoding experiment. The corresponding PSNR of the original video is 33.85 dB. For bit rate reduction, we have 5 different versions with the corresponding bit rates decreasing from 500 kb/s to 100 kb/s with a 100 kb/s interval. Normalizing the CPU load for a full transcoding session as 1 unit, only consuming 0.4 for all bit rate reduction cases. The corresponding metadata size is 10% of the original video file size, regardless of bit rate differences.

For frame rate transcoding, we experiment with 5 different qualities as well. They are 500 kb/s and 400 kb/s at 30 f/s, 300 kb/s and 200 kb/s at 15 f/s, and 100 kb/s at 5 f/s. Normalizing the CPU load for a full transcoding session from 30 f/s to 15 f/s as 1 unit, the CPU load for full transcoding from 30 f/s to 5 f/s and from 15 f/s to 5 f/s are 0.44 and 0.36 unit, respectively. For meta-transcoding, the corresponding CPU loads are 0.5, 0.27, and 0.19 unit and the corresponding metadata size is 20%-25% of the original video file size for transcoding between different frame rates, depending on the source bit rate and target frame rate.

With these parameters, we evaluate PAT over a topology of 1000 nodes. Specifically, one source node has all the different versions and its bandwidth capacity is 5 Mb/s. The distribution of bandwidth capacity for the rest of the nodes is as follows: 5Mb/s (11%), 2Mb/s (3%), 896kb/s (9%), 384kb/s (21%) and 256kb/s (56%). Assuming the version index 1 representing the original version with increased index indicating lower quality versions, we further dictate the nodes with 5Mb/s and 2Mb/s connections to randomly request versions 1 to 3, and nodes with other bandwidth capacities to randomly request versions 1 to 5. The CPU constraints of nodes randomly are distributed from 1 unit to 2 units. The system starts with one source node capable of serving all versions and 1000 nodes dynamically join and
depart from the streaming service. Their arrival pattern follows a Poisson distribution with the mean arrival rate as 1 request per second and the maximum arrival interval of 10 seconds. Nodes depart randomly after receiving the service for a duration ranging from 250 seconds to 1000 seconds.

4.2 Experimental Results

We evaluate the performance of our proposed scheme (PAT-Meta) by comparing it with two other schemes. No-Transcoding indicates the system in which no node can perform transcoding but the source node can serve precoded versions and a joining request is accepted if the requested or a lower quality version is available. PAT-Normal indicates the PAT system in which all transcoders perform full transcoding without assistance from metadata.

4.2.1 Effect on the Client Received Video Quality

First, we compare the video quality received at the peers. Figure 2 and Figure 3 show the number of nodes that are rejected or accepted but receiving lower than desired quality versions as a function of time for bit rate transcoding and frame rate transcoding, respectively. The plots are based on a 20-second window in time. Both transcoding schemes in PAT significantly outperform No-Transcoding, and PAT-Meta performs better than the PAT-Normal scheme. Overall, about 30% nodes could not receive their desired video qualities in No-Transcoding, while it is only about 4% in PAT-Meta.

Figure 4 and Figure 5 further plot the average PSNR of the received video at each node sorted in increasing PSNR values. To simplify evaluations, we assume a PSNR of 15 dB for nodes that are rejected when joining. For both experiments, PAT-Normal and PAT-Meta achieve better overall quality than No-Transcoding. Comparing the number of rejected nodes (PSNR at 15 dB), No-Transcoding rejects much more nodes than PAT-Normal and PAT-Meta.

Table 1 summarizes the average video quality that clients receive in two experiments. Overall, the average video quality experienced by clients in PAT-Meta is the best. Although PAT-Normal achieves similar quality gain over No-Transcoding, it is at the cost of much more computing overhead as shown next.

<table>
<thead>
<tr>
<th></th>
<th>No-transcoding</th>
<th>PAT-Normal</th>
<th>PAT-Meta</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
<td>28.14</td>
<td>30.86</td>
<td>31.49</td>
</tr>
<tr>
<td>Frame</td>
<td>27.19</td>
<td>30.54</td>
<td>30.82</td>
</tr>
</tbody>
</table>

4.2.2 Effect on CPU and Bandwidth Consumption

Figure 6 and Figure 7 show the normalized CPU load required for bit rate and frame rate transcoding, respectively. Again, the CPU load is computed based on a 20-second window in time. As expected, there is no CPU consumption for transcoding in No-Transcoding. With the assistance of metadata, PAT-Meta significantly outperforms PAT-Normal. On average, PAT-Meta can save 58% and 39% CPU load comparing with PAT-Normal for bit rate transcoding and frame rate transcoding, respectively. In addition, frame rate transcoding requires generally more CPU load than bit rate transcoding, which is consistent with our previous analysis.

PAT-Meta significantly reduces the CPU load required for transcoding by sharing metadata in the metadata overlay, which leads to some traffic overhead. To evaluate this traffic overhead, Figure 8 shows the total traffic for bit rate transcoding, and Figure 9 shows the corresponding result.
for frame rate transcoding. The traffic amount is summed every 20 seconds. With transcoding, both PAT-Normal and PAT-Meta serve better quality video and a larger number of clients than No-Transcoding, leading to a significantly increased total amount of traffic over No-Transcoding. The difference between the total traffic in PAT-Normal and PAT-Meta mostly indicates the additional bandwidth consumed for metadata sharing, which only accounts for at most 6% of the total traffic. Jointly considering the quality improvements and the computing and traffic overhead, we can see that PAT-Meta achieves the best quality improvement with least computing overhead and negligible traffic overhead.

5. CONCLUSION

To assist overlay streaming to a mixture of client devices, in this paper, we propose to build PAT that enables effective online transcoding in overlay streaming systems by leveraging node cooperation without requiring additional infrastructure support. To minimize the computing overhead, PAT effectively leverages the meta-transcoding with a metadata overlay that can instantly share the metadata with other transcoding nodes. Driven by parameters obtained from practical transcoding and meta-transcoding schemes, our simulation shows the effectiveness of our proposed scheme.

In PAT design, we assume the peers in the system are trusted and cooperative. We plan to investigate the security issues when untrusted/malicious peers present.

6. ACKNOWLEDGMENTS

We would like to thank William L. Bynum and anonymous reviewers for their helpful comments on this paper. The work is supported by NSF grants CNS-0509061 and CNS-0621631.

7. REFERENCES

Multidimensional Transcoding for Adaptive Video Streaming

Jens Brandt  Lars Wolf
IBR
Technische Universität Braunschweig
Mühlenpfordtstrasse 23, 38106 Braunschweig, Germany
{brandt|wolf}@ibr.cs.tu-bs.de

ABSTRACT
Video content providers often offer their video streams at three or more different quality levels: low, medium and high quality. These different levels are needed to support different target devices. However, this static kind of adaptation to discrete quality levels cannot meet the requirements of all existing multimedia devices. Because of the increasing heterogeneity of Internet-enabled video devices a more dynamic and individual adaptation to the requirements of the consuming device is needed. One approach for video adaptation is the use of video transcoding techniques, where modifications of the video stream are done in the compressed domain to save processing power. Many transcoding approaches for digital video can be found in the literature, but these focus on adaptation in one single dimension only, such as temporal adaptation, spatial adaptation or quality adaptation. To support as many video-enabled devices with different capabilities as possible, adaptation in more than one dimension is needed. In this paper we present a multidimensional transcoding approach for MPEG-4 encoded video, which smartly combines existing transcoding techniques to enable fine grain adaptation for different video devices.

1. INTRODUCTION
The prominence of digital video on the Internet is rising constantly. But transmission as well as decoding of digital video streams still has high resource requirements. To meet the requirements of different target devices, video content providers have to offer their video streams at different quality levels. Due to the increasing heterogeneity of Internet-enabled video devices a static adaptation cannot meet the requirements of all devices. Especially when looking at mobile devices, a fine grain adaptation to the capabilities of the device is mandatory. The possibilities of video playback with mobile devices are mainly restricted by the limitation of the following resources: network bandwidth, processing power, display resolution, memory size and remaining energy. Although mobile video devices are getting more and more powerful, the processor speed of current PDAs, smart phones or pocket multimedia players is only about 300 – 600 MHz and their memory size is about 64 MB. Without special video decoding hardware such devices are often incapable of decoding and displaying high quality video streams from the Internet. This problem will remain in the near future, also since quality demands are rising as well.

Table 1 shows the impact of different video coding parameters i.e., spatial, temporal and detail resolution, on the resource requirements which the decoding device has to satisfy. In this table as well as in the rest of this work we focus on block-based video codecs that use motion compensation (MC) and discrete cosine transfer (DCT) for video compression. Examples of such video coding schemes are MPEG-1/2/4, H.261, H.263 and H.264. All three parameters that are given in the table directly affect the number of coded bits in the stream and therefore also affect the resource requirements on the decoding device. Especially the spatial resolution has an impact on all four resources. When reducing the spatial resolution of a video stream, its bit rate is reduced, which in turn also reduces the necessary network bandwidth and processing power to decode the video. Furthermore, because of smaller frame resolutions, the amount of memory needed to buffer reference frames as well as the consumed energy of the device are also reduced. Mobile devices often have a small screen resolution, thus the spatial resolution needs to be tailored to the screen resolution anyway. Further bit rate reduction can be achieved by reducing the temporal resolution (frame rate) and decreasing the detail resolution (increasing the quantization factor). Because of the reduced number of frames and the reduced number of coded DCT values, both reductions also have an impact on the processing power needed for decoding. Additionally, all parameters have an impact on energy consumption. In [10] it was shown that the energy consumption of video decoding can be reduced to 75 – 58% by increasing the quantization level which results in a quality decrease of just 5 – 13%.
In the literature mainly two different approaches for video adaptation can be found: scalable video coding and video transcoding. With scalable video coding the video stream is separated into a base and one or more enhancement layers. The base layer contains the complete stream in a low quality and each additionally received layer increases the quality of the video. If an adaptation is needed, those layers exceeding the capabilities of the device can be discarded, either on the device itself or in the network. In our work we concentrate on single layer video streams, because of the lower complexity needed for their encoding and decoding compared to multi layer video streams. Furthermore, most of the available video streams on the Internet are single layer videos.

Video transcoding is a well-known mechanism for the adaptation of video coding parameters which are defined at encoding time. Compared to video encoding, where the video is completely decoded before it is encoded again, video transcoding works in the compressed domain. Thus, processing time can be saved at the expense of lower flexibility. Several specialized transcoding mechanisms exist, but hardly any of them focuses on transcoding in more than one dimension. In this paper we present a novel architecture for multidimensional transcoding, which we have implemented for MPEG-4 video streams. This architecture is intended to be used in our multimedia gateway system to support adaptive video streaming for mobile clients, which we have already presented in previous work [2].

The remainder of this paper is organized as follows: In section 2 we give an overview of existing transcoding techniques and related work. Afterwards we present our processing architecture for multidimensional transcoding in section 3. In section 4 we describe our multidimensional transcoding approach and implementation for MPEG-4 video, based on the aforementioned architecture. The evaluation of our implementation is presented in section 5. Finally we conclude this paper in section 6.

2. VIDEO TRANSCODING

The field of video transcoding has been explored in several recent studies. Most of them focus on different specialized transcoding techniques. A comprehensive overview of the research area dealing with those transcoding techniques is given in [13] and [7]. However, almost all of the presented approaches in the literature focus on adaptation of one single parameter, such as the three aforementioned, the bit stream syntax or the semantics of the content. Those coding parameters can also be interpreted as different dimensions in which the input video can be adapted. To handle the huge variation of resource constraints and users’ preferences a multidimensional adaptation is needed. Adaptation along the spatial dimension is used to tailor the resolution of a video stream to the display resolution of the requesting device. This also reduces the memory and processor requirements on the device. Adaptation in the temporal dimension as well as adaptation in the dimension of detail resolution can be used for further bandwidth reduction.

Due to the compressed nature of digital video, the pixels of the video frames cannot be manipulated directly. Additionally, the reduction of the frame rate is not trivial, because of the dependencies between consecutive frames, caused by motion compensation. In digital video only some frames are encoded as complete frames i.e., key frames. The majority of the encoded frames use previous and future frames as reference frames, so that only the differences to those frames need to be encoded (i.e. P- and B-frames). The most obvious way to adapt compressed videos is the use of the cascaded pixel domain transcoder (CPDT), which simply consists of a cascaded decoder and encoder. In fact, the CPDT is a video recoder as mentioned before. But a complete decoding and encoding is quite inefficient. Moreover, the information that is already available from the previous encoding should be reused for transcoding to enhance the produced quality of the video. Most video coding standards such as MPEG video use the discrete cosine transfer (DCT) to transform each frame from the pixel domain into the frequency domain. In a CPDT this transformation needs to be performed from the frequency domain and back, which is computationally very intensive. The main idea of video transcoding is the avoidance of this complete decoding and encoding. Mathematically, the DCT is a linear and orthogonal transformation which is distributive with respect to matrix multiplications. To avoid the computation of the DCT we can take advantage of these characteristics of the DCT [3]: To manipulate a video block A we need two matrices X and Y to compute the manipulated block B = X . Y . Due to the linearity of the DCT we have:

$$DCT(B) = DCT(X . A . Y) = DCT(X) . DCT(A) . DCT(Y)$$

Thus we can realize every linear pixel manipulation in the frequency domain. However, in most digital video formats motion compensation is used to utilize temporal dependencies between frames for compression. These existing dependencies are encoded by motion vectors which are pointing to a block in a reference frame and by the resulting error block, i.e. the difference between the current and the referenced block. Thus, to be able to manipulate a coded block in the frequency domain we have to compute the inverse motion compensation (IMC) in the frequency domain, which was firstly presented in [3] and subsequently optimized in further publications, for instance in [12] and [9]. These mechanisms were used in several proposed transcoding architectures, such as a frequency domain requantization transcoder proposed in [1], a frame-skipping transcoder proposed in [5] or in a spatial resolution downconversion transcoder in [8] and [6]. The only approach we are aware of that addresses transcoding in more than one parameter was proposed by Shanableh et. al. in [11]. They focused on heterogeneous transcoding of MPEG-1 video into H.261 / H.263 video with spatial and temporal resolution reduction. However, the reduction of each single parameter was treated separately and no architecture for the combined transcoding of the examined parameters was presented. In summary, an integrated approach for multidimensional video transcoding, especially for mobile devices has not yet been proposed.

In this work we concentrate on transcoding in three different dimensions which we have identified to be the most important for video adaptation for mobile devices: spatial resolution, temporal resolution and detail resolution. By transcoding in these three dimensions we can support fine grain video adaptation for mobile devices. However, our approach is not limited to these dimensions but can easily be extended to support adaptation also in other dimensions, such as heterogeneous transcoding from one video format to another.
3. PROCESSING ARCHITECTURE

Starting from the studies of different transcoding mechanisms for video adaptation in one single dimension, we have developed a multidimensional transcoding architecture. This architecture is based on the idea of smartly combining one-dimensional transcoders into a transcoder chain, which in turn forms a multidimensional transcoder. As illustrated in figure 1, in this architecture a transcoder $T_i$ is a module which can consume and produce video frames. Such a transcoder consumes a frame only if another transcoder or program module requests a frame from it. In other words, each transcoder chain needs exactly one active component which controls the whole transcoding process. In the figure this is done by the controller, which is also responsible for the setup and configuration of the whole transcoder chain. Therefore, each transcoder in the chain can generate events $e_i$ to notify the controller about its current state. In order to transcode a video stream by the shown chain, the controller requests a frame from the last transcoder in the chain ($T_2$) which consumes the frame from $T_1$ which reads the original stream from one or more video sources. Examples of such sources are files, network streams or streams from a live video source. The frames which the controller requests from $T_2$ are written to an output stream. The transcoder $T_2$ in the figure is a combination of two other transcoders $T_{21}$ and $T_{22}$.

![Transcoder Architecture - Overview](image)

**Figure 1: Transcoder Architecture - Overview**

The video frames passed from one transcoder to another are typically not completely decoded, but they may be partially decoded. To indicate the actual coding state of a frame we have defined the following five states: **encoded**, **partially decoded**, **quantized**, **dequantized** and **IMC**. These states will be described in detail in the following section. If two or more transcoders are connected to setup a transcoder chain, the states of the frames produced by each transcoder have to match to the frame states, which its successor in the transcoder chain accepts.

This modular processing architecture provides good flexibility as well as good control of the whole transcoding process. During this process, each transcoder can generate events ($e_i$ in the figure) to notify other program modules, e.g. the controller. Program modules can listen to these events and can firstly react to and secondly control each step of the transcoding process. These events are especially needed to inform the controller about the status of each frame, so that it can fully control each transcoding step of the frame.

4. MULTIDIMENSIONAL TRANSCODING

Since the focus of our research is dynamic video adaptation for mobile devices, we developed a multidimensional MPEG-4 transcoder for mobile devices, which uses the processing architecture presented in the previous section. Mobile devices often have a small screen resolution, which makes the use of spatial adaptation mandatory. Fortunately, this also reduces the bit rate and therefore the required bandwidth of the video stream. In order to achieve further reduction of the bandwidth of the video we decided to tailor the frame rate by the use of temporal adaptation and finally adjust the bandwidth of the stream by changing the quantization factor. The Order in which the video should be transcoded in the three dimensions is quite obvious: Firstly those frames which are not needed in the resulting video should be dropped. Secondly the resulting frames should be scaled to the designated resolution and finally the DCT values of the remaining frames should be quantized again.

A simple combination of the existing transcoding mechanisms is not efficient. Some steps, such as the inverse motion compensation (IMC), are needed in all transcoders and therefore should be computed only once. While some transcoding mechanisms in the literature may not need the IMC, we have identified that in the case of multidimensional transcoding the IMC is always needed. For instance, when dropping P-frames new motion vectors have to be computed. Additionally, the error blocks coded in the remaining frames have to be recalculated by the use of this new motion vectors. Therefore, when working in the compressed domain, the DCT values of macro blocks of different frames has to be added. However, in MPEG-4 each macro block may have been encoded with its own quantization factor, which makes a direct addition of different macro blocks impossible. Moreover, when changing the DCT values of a frame which is used as a reference frame, the decoding of the referencing frame becomes inaccurate, because the encoded error blocks in the referencing frame were calculated based on the original DCT values of the reference frame. This inaccuracy is called drift error and can be avoided by using a drift free loop, which performs the IMC for each inter-coded block and eliminates the occurring error from the current block.

Only in the case of B-frame dropping the IMC can be avoided, because B-frames are never used as reference frames and therefore can be dropped without any influence on other frames. Thus, to drop a B-frame we only need to decode the type of the frame. Afterwards we can compute the IMC for the remaining frames which removes any temporal dependencies between the frames. However, the motion vectors are still saved in each frame for future use. Now, we can go on with dropping P-frames.

When dropping P-frames, motion information gets lost and motion vectors of subsequent P- or B-frames become invalid. Thus, we need to define new motion vectors for the following frames. Instead of searching new motion vectors, we can use one of the mechanisms for redefining motion vectors described in the literature, which were already mentioned in section 2. As the number of frames has already been reduced we can now reduce the spatial resolution of the remaining frames by using an existing spatial reduction method from the literature, e.g. the one proposed in [4]. When downsampling a video frame all motion vectors of this frame (in case of a P- or B-frame) get invalid, because they are still pointing to the original region in the reference frame. Thus, also the coefficients of the motion vectors have to be downscaled. Ad-
ditionally, we have to compute one new macro block from several existing macro blocks, because macro blocks have a fixed resolution (e.g. 16 × 16 Pixel in MPEG video). But also the number of motion vectors of a macro block is fixed and we may have to create one downscaled motion vector by interpolation of several existing vectors. In the case of MPEG-4 we can benefit from the ability of MPEG-4 to use four vectors per macro block. For instance, if a video is downscaled by a factor of two and if each macro block has exactly one motion vector, we do not need to interpolate them but can use all of them in the new macro block. Afterwards we have to compute the motion compensation (MC) by using the newly created motion vectors. Within the MC we can either use the same quantization factor as before or a new one for further compression.

4.1 Frame Decoding

For communication between the transcoders in the chain, we have created a data structure which contains all information about the frame as well as its data. The encoding state of the frame indicates how much of the frame has already been decoded. Frames in the state encoded are completely encoded frames, as they are read from the input stream. After parsing the first few bytes of a frame, we get some information such as the type of this frame, and its state changes to partially decoded. When we further decode the frame, we get the quantized values of the DCT coefficients and the state changes to quantized. The next step of decoding is to dequantize the DCT values, which changes the state of the frame to dequantized. The final decoding step needed for transcoding is the computation of the IMC, after which the state of the frame is changed to IMC. Figure 2 shows the state chart of the frame states.

![Figure 2: Frame States](image)

4.2 Transcoder Chain

The complete transcoder chain is shown in figure 3. Not only are the modules which effectively transcode the video frames implemented as a transcoder but also those modules which are needed to pre- and post-process the video frames. At the beginning of the chain the PartialDecoder reads the video stream from the input and creates the data structure for each frame. Only the information needed to identify the frame type is decoded and saved in the data structure. Because dropping of B-frames does not affect any other frames, the BFrameDropper is placed right behind the PartialDecoder. It can read the type of a frame and drop those frames which are identified as B-frames. Afterwards the StreamDecoder is used to decode the quantized DCT values as well as the motion vectors (in the case of intra-coded frames) of each frame. The Dequantizer computes the inverse quantization of the DCT values and the IMC module is used to compute the inverse motion compensation. After the IMC the video frame contains all macro blocks in the frequency domain without any temporal dependencies. The PFrameSkipper can skip P-frames now and the Scaler can reduce the spatial resolution of the frames. As mentioned before, both of them has to redefine the motion vectors of the remaining frames. The MC component implements the drift free loop to avoid drift errors and computes the motion compensation. There is no separate requantization transcoder in the chain, because requantization is already integrated in the drift free loop of the MC module. After the motion compensation has been computed, the frames are encoded into a video stream by the Encoder.

4.3 Implementation

We have created a prototype implementation of our proposed multidimensional transcoder, written in C++. This implementation is intended to be used for MPEG-4 video adaptation for mobile devices. For bit stream parsing and decoding of the video frames we have created a namespace called m4v, which contains all MPEG-4 specific classes. Classes implementing transcoding techniques are placed in a separate namespace called trc.

The frame-skipping and scaling transcoders are implemented in a way that they are independent of the video format which they process. Thus, they can also be used for other video coding formats, such as MPEG-1/2 or H261/3. To support such video formats, our implementation can be extended easily by adding new decoder and encoder modules which are capable of parsing the entire video streams.

The PFrameSkipper skips a frame whenever the controller requests this. It uses the forward dominant vector selection presented in [14] to find new motion vectors of following frames. The HalfScaler uses bilinear interpolation in the frequency domain as presented in [4]. In our current implementation we are able to downscale the spatial resolution by a factor of two. Thus, we have to compute one down-scaled macro block from four incoming macro blocks. If each incoming macro block has only one motion vector, we can simply use those four vectors, each downscaled by a factor of two, in the downscaled macro block. Otherwise, if at least one of the incoming macro blocks has already four motion vectors, we use the vector of the most active block. The activity of a block is computed by the sum of all DCT coefficients. The module MC computes the motion compensation by using the redefined motion vectors and may use the original or a new quantization factor, depending on the request of the controller.

For testing purposes of our transcoders, we have also created a test application, which is a kind of a frontend to our transcoding architecture implementation. This application, which is written in Java, can be used to load a reference video in rawvideo format as well as a source MPEG-4 video. Furthermore a transcoding chain can be chosen to be used for transcoding. When starting the test, the source video is transcoded by the chosen transcoding chain and all three

![Figure 3: Transcoder Chain](image)
videos (reference, source and transcoded) are displayed on the screen. Additionally, the PSNR values of each frame are calculated and displayed in a graph. A screenshot of this application can be found in figure 4.

5. EVALUATION

For the evaluation of our implementation we selected two different videos. The first video, which is titled “Akiyo”, shows a woman presenting some news. The background is quite static and there is not much motion in this video. The second one is called “Foreman” and there is more motion in the video. A foreman is talking into the camera and is gesticulating with his hands. In the last third of this video, the camera turns to the right and the shell of a building in the background becomes visible. Both videos have 300 frames in CIF (352x288) resolution, a group of video object panes (GOV) length of 30 and a frame rate of 25 fps. They were encoded to MPEG-4 by the use of the XviD codec and a fixed quantization level of one ($q = 1$). The Akiyo sequence has a bit rate of 512 KBit/s and the Foreman sequence a bit rate of 1024 KBit/s.

For both test videos we evaluated the PSNR values of the transcoded version compared to the recoded version of the sequence. The temporal as well as the spatial resolution were halved (QCIF, 12.5 fps) and the quantization factor was increased. The recoded version was produced by the use of the XviD codec and the transcoded version by the use of our implementation. The PSNR values were computed by upsampling each decompressed frame of the transcoded and recoded video respectively and using the uncompressed CIF frames of each video for reference. Figures 5 and 6 show the results of the transcoded video compared to the recoded one for both the Akiyo and the Foreman sequence. In both cases the PSNR values of the transcoded versions are higher than those of the recoded version. Because of the medium amount of motion in the Foreman sequence, the transcoded version benefits from the reuse of existing motion vectors and thus the PSNR differences between the recoded and the transcoded version are significantly higher than those of the Akiyo sequence. For the transcoded videos two different quantization factors are shown, because the bit rate of the transcoded version is a little bit higher than that of the recoded video when using the same quantization factor. Thus to compare the quality of the videos with the same bit rate we included the PSNR values of the transcoded video with a higher quantization factor in the figures. Additionally, we evaluated the achieved quality of our transcoder for the complete range of allowed quantization factors ($q = 2 - 31$). In figure 7 the average PSNR values of transcoded and recoded versions of the videos are shown. For lower bit rates our transcoder produces better or comparable PSNR values for both videos to the XviD codec. There are two outliers in
6. CONCLUSION & FUTURE WORK

In this paper we have presented a new and flexible processing architecture for multidimensional video transcoding. We have identified three transcoding dimensions which are necessary for fine grain adaptation for mobile devices. Based on our architecture we have developed an integrated transcoder which transcodes an incoming video stream within the aforementioned three dimensions. The evaluation of our implementation shows promising results, with slight improvements of the produced quality with regards to PSNR values of the video frames. At the moment our implementation is not optimized as other video codecs on the market. Thus, the processing time needed for transcoding is quite high. But the evaluation results of the achieved quality are promising and we will optimize our implementation in the future. Additionally, by using advanced MPEG-4 features we can reduce the produced bit rate without affecting the quality of the stream and thus produce better quality results also for higher bit rates. Further, we plan to integrate an intelligent rate control mechanism into the controller in order to control the quality and bit rate of the transcoded video in the best possible way. Based on our transcoder implementation we further plan to evaluate the produced quality not only in terms of PSNR values but also by the use of subjective tests with potential users.

7. REFERENCES


![Figure 7: Average PSNR values](image)

Table 2: Transcoding Runtime Performance

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Processing Time</th>
<th>Frame Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Akiyo</td>
<td>7.092 s</td>
<td>21.15 fps</td>
</tr>
<tr>
<td>Foreman</td>
<td>16.273 s</td>
<td>9.22 fps</td>
</tr>
</tbody>
</table>

Table 2: Transcoding Runtime Performance
Optimal Partitioning of Fine-Grained Scalable Video Streams

Cheng-Hsin Hsu and Mohamed Hefeeda
School of Computing Science
Simon Fraser University, Surrey, Canada
{cha16, mhefeeda}@cs.sfu.ca

ABSTRACT

The increased popularity of video streaming over the Internet attracts numerous clients. These clients are quite heterogeneous in terms of network bandwidth and processing capacity. To accommodate this heterogeneity, fine-grained scalable (FGS) coding of video streams has been proposed in the literature. FGS streams are composed of two layers: base layer, which provides basic quality, and a single enhancement layer that adds incremental quality refinements proportional to the number of bits received. The base layer uses nonscalable coding which is more efficient in terms of compression ratio than scalable coding used in the enhancement layer. Thus for coding efficiency larger base layers are desired. Larger base layers, however, disqualify more clients from getting the stream. In this paper, we study and quantify the trade-off between the coding efficiency and the range of clients that can be supported. Then, we design an efficient algorithm to compute the optimal size of the base layer that will yield the best video quality for a given client distribution. We implement our algorithm and apply it on video sequences with different characteristics. Our experimental results show that our algorithm improves the average perceived quality for all clients.

1. INTRODUCTION

Video streaming over the Internet is increasingly getting very popular as higher bandwidth links and more powerful machines are becoming more affordable for end users. Users typically seek the highest possible video quality. Users, however, are quite heterogeneous in terms of network bandwidth and processing capacity. A conventional nonscalable coded stream only supports one decoding rate, which is insufficient in such a heterogeneous environment. This is because supporting clients with different bandwidth requires storing and serving multiple versions of each video stream. To cope with this heterogeneity, various scalable coding techniques have been proposed in the literature. A scalable coded stream consists of various representations of the original video sequence, with different resolutions, frame rates, or quality.

Scalable coders are roughly categorized into two classes: coarse-grained scalable and fine-grained scalable. Coarse-grained scalable (CGS) coders divide a video stream into multiple layers. They provide limited rate scalability at the layer level: clients receiving incomplete layers cannot use them to enhance quality. In contrast, fine-grained scalable (FGS) coders provide finer rate scalability and better error resiliency [8, 10, 13]. An FGS encoder compresses video data into two layers: a base layer which provides basic quality, and a single enhancement layer that adds incremental quality refinements proportional to the number of bits received. Arbitrary truncation (at the bit level) of the enhancement layer to achieve a target rate is possible for FGS coding. This in turn enables streaming servers to fully utilize available bandwidth of individual clients, which results in better video playback quality and ultimately higher user satisfaction.

The fine rate scalability of FGS, however, comes at an expense of coding efficiency. That is, an FGS coded stream results in lower quality compared to a nonscalable coded stream when both streams are reconstructed at the same bit rate. Previous research indicates that this coding efficiency gap is up to 2 dB in MPEG-4 FGS coders [14]. The two main causes of this coding efficiency gap are: (i) less accurate motion compensation as only base layer is used for motion estimation, and (ii) un-exploited correlation between base layer and enhancement layer. The coding efficiency gap is less significant in video sequences with low temporal redundancy since motion compensation does not provide much quality gain for these sequences. Furthermore, lower base layer rates result in larger gap because less information is contained in the base layer in this case, which leads to higher motion estimation error [10, 14].

While the temporal correlation is fixed for a given sequence, the base layer rate is a configurable parameter. Therefore, content providers may code a sequence at higher base layer rate to reduce the coding efficiency gap and strive for higher quality. This may increase perceived quality for some clients, which could allow the provider to charge higher service rates. On the other hand, a higher base layer rate may disqualify other clients from receiving the complete base layer stream. Since the base layer is nonscalable, these disqualified clients can not even render basic quality and effectively they are denied access to the video stream. Hence, there is a trade-off between coding efficiency and supported rate range, which can be gauged by the FGS base layer rate. More importantly, the average perceived quality for all clients depends on this trade-off, and thus depends on the choice of base layer rate.

To FGS encode a given video sequence, content providers have many options for the base layer rate. Each base layer rate determines the average perceived quality for all clients, which can be used as a metric for user satisfaction. Users who receive higher-quality video are willing to watch more programs or pay higher subscription fees. Thus, content providers need to maximize the average perceived quality to increase their profits. Unfortunately,
there are no systematic ways in the literature to aid content providers in choosing the optimal structure of FGS coded streams that would maximize the average quality for all clients.

In this paper, we first study and quantify the trade-off between the coding efficiency and the range of clients that can be supported. Then, we formulate an optimization problem to determine the base layer rate that achieves the best average video quality for a given client distribution. We design an efficient algorithm to solve this optimization problem. We implement our algorithm and apply it on video sequences with different characteristics. Our experimental results show that our algorithm improves the average perceived quality for all clients.

The rest of this paper is organized as follows. In the next section we summarize the related work. In Section 3, we present our analysis of FGS streams, formulate the optimization problem, and present our algorithm to solve it. We evaluate our algorithm in Section 4, and we conclude the paper in Section 5.

2. RELATED WORK

The coding efficiency gap of MPEG-4 FGS coders are studied in [10, 14]. The authors investigate the relationship between the FGS coding efficiency gap and the video temporal correlation. They found that the correlation coefficient between an enhancement layer frame and its motion-compensated reference frame is a good indication of the FGS coding efficiency. We study the efficiency gap of the state-of-the-art H.264 coders. Streaming systems [2, 6, 7, 9] account for the coding efficiency using a layering overhead, which represents the bit rate that does not contribute toward the video quality. Similarly, we model this overhead by a coding efficiency gap function, and we empirically estimate this function.

The performance of layered streams versus nonscalable streams is studied in [11]. The authors formulate a dynamic programming problem to compute the rate of each layer such that the average perceived video quality is maximized. The square root rate-distortion model [1] is used to estimate the coding efficiency of the layered coding. In [9], the authors consider broadcasting multi-layer video streams in a wireless cellular system with a given number of channels and client capacity distribution. They determine the optimal rate of each layer to maximize the average perceived quality. Unlike our work, these two works target coarse-grained streams which provide limited flexibility compared to fine-grained streams.

The authors of [16] study multicast streaming systems with many receivers. They partition receivers into several groups to maximize a system-wide utility function. A video stream used in such systems can be encoded into multiple cumulative layers. Several versions with different rates of the same stream can also be created. This work does not consider fine-grained streams, nor does it account for the layering overhead. Several papers have approximated layering overhead for performance comparison of layered streams and multiple version streams. For example, the work in [7] proposes a linear layering overhead function, which is inspired by the experimental results in [10]. In [2], simulation using the MPEG-2 two-layer scalable structure is employed for such comparison, where the layering overhead is assumed to be proportional to the enhancement layer rate.

3. PROBLEM FORMULATION

In this section, we first study the characteristics of FGS-coded streams highlighting the trade off between the coding efficiency and the range of clients that can be supported. Then, we formulate and solve the quality optimization problem considered in this paper.

3.1 Characteristics of FGS-Coded Streams

A fine-grained scalable (FGS) video stream is composed of two layers: base layer and enhancement layer. As depicted in Fig. 1, the base layer is nonscalable and must be received in its entirety to provide basic quality, while the enhancement layer can be truncated at arbitrary bit positions. Therefore, an FGS-coded stream can support a wide range of streaming rates, and thus many heterogeneous clients. Let us denote the bit rate of the base layer as $r_b$, and the maximum bit rate of the video stream as $r_{max}$. $r_{max}$ corresponds to the maximum possible quality of the video stream, and it is specified by the resources (storage and bandwidth) allocated to the video stream by the system administrator. An FGS-coded stream can be served at any bit rate $r$, where $r_b \leq r \leq r_{max}$.

Our problem in this paper is to determine the best base layer rate $r_b$ so that the average quality is maximized for all clients. To solve this problem, we need to study the implications of varying $r_b$. We design the following experiments to analyze these implications. We use the Joint Scalable Video Model (JSVM) reference software version 8.0 [5] in our experiments. A brief description of this software and how we configured it is given in Section 4. We choose two standard video sequences: City and Mobile, both in CIF format with 30 frames per second. We set $r_b$ at a specific value and encode the whole stream with a maximum rate $r_{max} = 3000$ kbps. Then we determine the quality that would be perceived by various clients decoding the stream at different rates. We consider clients in the range between 250 kbps and 3000 kbps with a step of 250 kbps. The quality is determined by decoding the stream and computing the peak signal to noise ratio (PSNR) in dB. We repeat the whole experiment for several values of the base layer rate, and for the two sequences. These are computationally intensive experiments and took many processing hours to complete.

The results of these experiments are presented in Fig. 2. Several observations can be drawn from this figure. First, the FGS streams have lower coding efficiency. For example, Fig. 2(a) indicates that decoding a nonscalable stream at rate 500 kbps results in 30 dB video quality, while decoding an FGS stream (with $r_b = 100$ kbps) results in 26 dB video quality. We define the quality gap $\Delta(r_b)$ as the quality difference between a nonscalable stream and a fine-grained scalable stream coded with base layer rate $r_b$. The quality gap can be explained by the additional overhead and un-exploited video redundancy caused by the scalable coding structure. A second observation we can make from Fig. 2 is that higher base layer rates lead to smaller quality gaps. For example, Fig. 2(a) shows that at decoding rate 1500 kbps, an FGS stream with $r_b = 750$ kbps results in about 1 dB quality gap compared to nonscalable stream, while an FGS stream with $r_b = 100$ kbps results in 6 dB quality gap. These differences can be explained by the fact that more temporal redundancy can be exploited if the base layer contains more information, i.e., is coded at a higher rate. This observation indicates that the quality gap $\Delta(r_b)$ is a non-increasing function of the base...
layer $r_b$. We further validate this property in the Section 4. We will use this non-increasing property in solving the quality optimization problem in the next subsection. Finally, we note that similar scalable coding inefficiencies were observed in MPEG-4 FGS coders [10]. This is consistent with our observations on the recent H.264 coders.

3.2 Problem Formulation

In this section, we formulate an optimization problem to find the base layer rate $r_b$ that achieves the highest average perceived quality for all clients. We consider heterogeneous client populations. We model this heterogeneity by dividing clients into $C$ classes. All clients belonging to the same class $c$ ($1 \leq c \leq C$) have the same bandwidth $b_c$. We assume that $b_1 < b_2 < \cdots < b_C$. The fraction of clients in each class $c$ is given by a probability mass function $f(b_c)$, where $\sum_{c=1}^{C} f(b_c) = 1$. No assumptions are made on the number of client classes or on the probability function. Without loss of generality, we assume that $b_C \leq r_{\text{max}}$. If otherwise, we combine classes with bandwidth smaller than $r_{\text{max}}$ in a class with bandwidth equal to $r_{\text{max}}$. We can do that because no matter how large the client bandwidth is, it cannot receive more than the maximum rate $r_{\text{max}}$.

We write the optimization problem $P$ that maximizes the average perceived quality as follows:

$$ P : \max_{r_b} \sum_{c=1}^{C} q(b_c) f(b_c), \text{ where } r_b \in [0, r_{\text{max}}], \quad (1) $$

where $q(b_c)$ is the quality (measured as PSNR in dB) achieved by clients in class $c$.

A naive approach to solve the above problem is to try all possible values for $r_b$ in the range $[0, r_{\text{max}}]$. This is very costly because FGS coders allow for too many possibilities for $r_b$. We propose a better solution that takes at most $O(C)$ steps. Our approach is enabled by the following lemma.

**Lemma 1.** An optimal solution $r_b^*$ for the base layer rate that maximizes the average perceived quality for all users can be found at one of the rates $b_c$, where $1 \leq c \leq C$.

**Proof.** Referring to Fig. 2(a), we can re-write the quality of the FGS stream $q(b_c)$ for clients in class $c$ as:

$$ q(b_c) = \begin{cases} 0, & b_c < r_b \\ q_{\text{nl}}(b_c) - \Delta(r_b), & b_c \geq r_b, \end{cases} \quad (2) $$

where $q_{\text{nl}}(b_c)$ is the quality achieved by the nonscalable encoder at rate $b_c$, and $\Delta(r_b)$ is the quality gap between the FGS and non-scalable streams as defined in the previous subsection. Notice that the quality for clients in any class $c$ is zero if these clients do not have enough bandwidth to receive the complete base layer, i.e., if $b_c < r_b$.

We divide the search range $[0, r_{\text{max}}]$ into non-overlapping intervals $(b_{c-1}, b_c]$, where $c = 1, 2, \ldots, C$ and $b_0 = 0$. Now assume that the optimal base layer rate $r_b$ occurs in an arbitrary interval $(b_{c-1}, b_c]$. Since all classes with $b_c \leq b_{c-1}$ receive quality of zero, the maximization problem becomes:

$$ \max_{r_b} \sum_{c=2}^{C} [q_{\text{nl}}(b_c) - \Delta(r_b)] f(b_c), \text{ where } r_b \in (b_{c-1}, b_c]. \quad (3) $$

Notice that the only term that depends on $r_b$ in the above equation is the quality gap $\Delta(r_b)$. Thus to maximize quality, we need to minimize $\Delta(r_b)$. Recall that in the previous subsection we argued that $\Delta(r_b)$ is non-increasing function of $r_b$, we validate this argument in Section 4. Since $\Delta(r_b)$ is non-increasing in the interval $(b_{c-1}, b_c]$, no point in that interval could make the quality gap smaller than $\Delta(r_b = b_c)$. Thus, an optimal solution for $r_b$ occurs at $b_c$.

The above lemma tells us that to find an optimal base layer rate $r_b$, it suffices to check only the rates $b_c (c = 1, 2, \ldots, C)$. A straightforward approach to implement this lemma is to compute equation (3) at $c = 1, 2, \ldots, C$ and choose the rate the corresponds to the maximum quality. This would require computing the summation at every iteration, which would make the time complexity of the algorithm $O(C^2)$. A better approach is to iteratively compute each term from $c = C$ towards $c = 1$, and every iteration only adds the difference in quality to the quality computed in the previous iteration. The difference $d$ in quality between class $c$ and

![Figure 2: The coding efficiency gap between FGS and nonscalable streams. The gap decreases as the base layer rate $r_b$ increases. Increasing $r_b$, however, limits the number of clients that can receive the stream.](image-url)
that a simple quadratic function can approximate the R-D function. We present a sample of our results in this section due to space limitations.

4.1 Setup

The Video Coding Experts Group (VCEG) and the Moving Picture Experts Group (MPEG), known as the Joint Video Team (JVT), are developing a scalable video coding (SVC) standard as an extension of H.264 standard [4]. The emerging SVC design is detailed in the SVC Amendment Working Draft [15] and Joint Scalable Video Model (JSVM) [12]. The JSVM reference software is provided by the JVT team to demonstrate an effective implementation that complies with the SVC standard. We use the JSVM reference software version 8.0 [5] in our experiments.

The JSVM reference software is implemented in C++, and contains several executables. We use the following executables: H264-AVCEncoderLibTest, BitStreamExtractor, H264AVCDencoderLibTest, and FixedQPEncoder. The H264AVCEncoderLibTest is a configurable SVC encoder that can compress a raw video file into a global stream. This global stream consists of many embedded substreams, which deliver lower quality video representations at lower rates. The global stream is stored as a file. The BitStreamExtractor tool extracts a user-specified substream from an existing global stream and stores it in a new file. Further stream extractions from this substream file are possible as the syntax and semantics of the global stream files and substream files are identical. The H264AVCDencoderLibTest is an SVC decoder that decompresses coded stream into a raw video file. Since the H264AVCEncoderLibTest does not implement rate control algorithm for a user-specified rate constraint, we have to use quantization parameter (QP) to gauge the resultant stream rate. The FixedQPEncoder is a tool that searches the proper QPs to satisfy rate constraints. It iteratively calls H264-AVCEncoderLibTest with estimated QP values, and stops when the resultant stream rate is within an acceptable range of the desired rate.

We choose two standard video sequences for our experiments: City and Mobile, both are in CIF format with frame frequency 30Hz. We encode them with the widely adopted IBBBPBBBP group of picture (GoP) structure. We first encode a sequence with single layer configuration using the FixedQPEncoder tool to get appropriate QP values for the target base layer rate. We then use the same QP values to code an FGS stream.

4.2 Average Quality Improvement

We have implemented our FGSOPT algorithm in Matlab. As mentioned in the introduction section, we are not aware of similar algorithms in the literature that optimize the average quality by controlling the base layer rate. Therefore, we compare the results of our algorithm to the results of heuristic methods. That is, we choose two reasonable rates for the base layer and compare the resulting quality against the quality produced by our algorithm. Indeed, there are too many other choices and we cannot cover all of them in our experiments. This is not really an issue because our algorithm is provably optimal, and the best that heuristic methods can do is to approach our algorithm by trial and error.

We choose ten classes and uniformly distribute clients among these classes. The bandwidth range for clients is between 0 and 3000 kbps. We run the FGSOPT algorithm to compute the optimal base layer rate. We choose two base layer rates for comparison: 100 and 1000 kbps. We compute the perceived quality for each client class and the average quality over all classes. The results are shown in Fig. 4. The figure clearly shows that the average quality over all classes has been improved using our FGSOPT algorithm.
4.3 Quality Gap Function

The FGSOPT algorithm assumes that the quality gap $\Delta(r_b)$ is a non-increasing function of base layer rate $r_b$. To validate the accuracy of this assumption, we compute the quality gap at various base layer rates. We use the reference software to encode the two test sequences with base layer rates between 100 kbps and 3000 kbps with an increment of 250 kbps. Each base layer rate results in a unique FGS coded stream that supports decoding rates between $r_b$ and 3000 kbps. To quantify the coding efficiency gap at a specific base layer rate $r_b$, we decode the stream at many decoding rates between $r_b$ and 3000 kbps and take the average over all of them. We compute the reconstructed quality at each decoding rate by first extracting the substream that matches this rate. Then we decode the extracted substream and compare it against the original video stream. A sample result is shown in Fig. 5. The figure confirms the non-increasing property of the quality gap function. On the same figure we plot a degree 4 polynomial function that best-fits the quality gap curve. We do this because the FGSOPT algorithm needs to compute the quality gap $\Delta(r_b)$ at different base layer rates. Thus instead of empirically measuring the quality gap at too many base layer rates, which is computationally expensive, we estimate the polynomial function and employ it in the algorithm. Estimating the polynomial function requires measuring the quality gap only at a few base layer rates.

Figure 4: Expected quality of individual client classes and the overall average quality for all clients when the base layer rate $r_b$ is set to three different values: the optimal computed by our algorithm (denoted by FGSOPT), and two other rates. Clients in even number classes are omitted for figure legibility.

Figure 5: The coding efficiency gap $\Delta(r_b)$ between FGS and nonscalable streams. The figure shows that $\Delta(r_b)$ is non-increasing function, and it can be modeled by a polynomial function with degree 4. Sample data shown for Mobile sequence.

Figure 6: The rate-distortion (R-D) function $q_{ns}(r)$ of nonscalable streams. The figure shows that a quadratic function provides a good approximation for the R-D function. Sample data shown for City sequence.
4.4 Rate-Distortion Function

The FGSOPT algorithm requires a rate-distortion (R-D) function that estimates the expected distortion at a given decoding rate when the stream is encoded in a nonscalable manner. Through extensive experiments, we have found that this R-D function can be approximated by a simple quadratic function. Fig. 6 shows a sample result, where we compute the R-D function at 12 sampling bit rates for City sequence. The figure also shows the best-fit quadratic function produced by the Matlab curve-fitting tool for the sample points. We note that the sample result in Fig. 6 provides guidelines for the administrator on the shape of the R-D functions and should be considered as a first approximation. Indeed, more elaborate R-D models can be found in the literature, but they are quite complex and expensive to implement. For detailed discussion and comparisons of various R-D models, see for example [3] and references therein.

5. CONCLUSION AND FUTURE WORK

In this paper, we first investigated the characteristics of FGS coded video streams. We designed several experiments using the emerging H.264/MPEG-4 SVC coder to study the trade-off between the coding efficiency and the range of clients that can be supported. The base layer rate is the main controlling parameter: Larger base layer rates yield higher coding efficiency but support fewer client classes, and vice versa. Our experiments show that the coding efficiency gap is a non-increasing function of the base layer rate. Then, we formulated an optimization problem to determine the base layer rate that achieves the best average video quality for a given client distribution. Solving this optimization problem is expensive, because there are too many possible choices for the base layer rate of FGS coded streams. To address this complexity, we proposed a simple algorithm that runs in linear time. We proved that our algorithm yields the optimal base layer rate. We implemented our algorithm and compared the FGS structures produced by it against two rule-of-thumb coding structures, which is the current practice. Our results indicated that our algorithm achieves better average perceived quality for all clients. Our proposed algorithm can be used in various applications. For example, given a long-term client distribution, our algorithm can aid content providers in choosing the optimal structure of stored FGS video streams. We designed several experiments using the Matlab curve-fitting tool for the sample points. The figure also shows the best-fit quadratic function produced by the Matlab curve-fitting tool for the sample points. We note that the sample result in Fig. 6 provides guidelines for the administrator on the shape of the R-D functions and should be considered as a first approximation. Indeed, more elaborate R-D models can be found in the literature, but they are quite complex and expensive to implement. For detailed discussion and comparisons of various R-D models, see for example [3] and references therein.

6. REFERENCES


When is P2P Technology Beneficial for IPTV Services?

Yih-Farn Chen, Yennun Huang, Rittwik Jana, Hongbo Jiang, Michael Rabinovich, Bin Wei, and Zhen Xiao

ABSTRACT
This paper studies the conditions under which peer-to-peer (P2P) technology may be beneficial in providing IPTV services over typical network architectures. It has two major contributions. First, we contrast two network models used to study the performance of such a system: a commonly used logical “Internet as a cloud” model and a “physical” model that reflects the characteristics of the underlying network. Specifically, we show that the cloud model overlooks important architectural aspects of the network and may drastically overstate the benefits of P2P technology by a factor of 3 or more. Second, we provide a cost-benefit analysis of P2P video content delivery focusing on the profit trade-offs for different pricing/incentive models rather than purely on capacity maximization. In particular, we find that under high volume of video demand, a P2P built-in incentive model performs better than any other model for both high-definition and standard-definition media, while the usage-based model generally generates more profits when the request rate is low. The flat-reward model generally falls in-between the usage-based model and the built-in model in terms of profitability.

Keywords: IPTV, P2P streaming, Content distribution network, FTTN, Video-on-Demand.

1. INTRODUCTION

Internet protocol TV (IPTV) promises to offer viewers an innovative set of choices and control over their TV content. Two major U.S. telecommunication companies, AT&T and Verizon, have invested significantly to replace the copper lines in their networks with fiber optic cables for delivering many IPTV channels to residential customers.

A viewer can receive IPTV videos in good quality if the available bandwidth satisfies the need of video encoding rate for the target resolution and frame rate. To provide sufficient bandwidth for IPTV services, Internet service providers use high speed xDSL or cable networks to deliver video content to viewers’ set-top boxes. As an example, AT&T Light-Speed is using Fiber-to-the-Neighborhood (FTTN) Networks. Its architecture consists of a small number of national super head-ends (SHE) and a large number of local video hub offices (VHO). The super head-ends serve as the national content aggregation points for broadcast and video on demand encoding. The local video hub offices provide aggregation and storage of local content. Each video hub office serves as a Video-On-Demand (VOD) library and distributes video content through local access switches to the customers.

We refer to this network hierarchy as the “physical” model throughout the paper. FTTN networks can provide 20-25Mbps bandwidth to each household, which is typically enough to support several high quality TV streams as well as high speed Internet and Voice over IP (VoIP) services.

A significant problem in providing IPTV services is its high deployment and maintenance cost. In addition, the capacity of the video servers can quickly become a bottleneck. One solution to alleviate the load on servers is to use peer-to-peer (P2P) systems like Skype [15] or Konkiki [10]. While early P2P systems were mostly used for file downloading, recently there have been several efforts on using the peer-to-peer approach to support live streaming [16][17][5][2][3][11] and VOD streaming[14][7][13][6]. Existing research studies that evaluate the benefits of P2P video content delivery typically do not consider the constraints of the underlying service infrastructure (e.g., [12][18]). Rather, they view the network as a “cloud”. Researchers, however, are increasingly aware of the need to reduce cross-ISP P2P traffic, while maintaining satisfactory P2P performance[4].

In this paper, we reveal the deficiency of this cloud model and investigate when P2P streaming can be beneficial in an IPTV environment. As we will see, P2P video sharing can be harmful under certain network conditions.

Another challenge for P2P streaming in an IPTV environment is the pricing strategy. Most broadband ISPs today charge a flat fee for providing bandwidth. Usage-based pricing has emerged in some markets but even in those cases it is limited to volume-based pricing. Among the limited early work on pricing strategies for P2P, Adler, et al. [1] provided a comprehensive model applicable to a variety of P2P resource economies. Implementation of peer selection algorithms in realistic networking models like the IPTV environment was not addressed. Hefeeda et al. presented a cost-
profit analysis of a P2P streaming service for heterogeneous peers with limited capacity [8]. The analysis shows that the service provider can achieve more profit by providing the appropriate incentives for participating peers. However, their analysis did not consider the bandwidth constraints of the underlying infrastructure and hence cannot be easily extended to our IPTV environment.

We make the following contribution in this paper:

- We compare two network models (the “cloud” model and the “physical” model) and show that the cloud model can dramatically overestimate P2P benefits by a factor of 3 or more.
- We couple three P2P pricing models (flat-fee, usage-based, and built-in) with a “physical” model and study their trade-offs from a profit perspective.

The rest of the paper is organized as follows. We describe the physical network model and constraints for the IPTV system in section 2. Section 2.3 provides the insights as to why a more accurate physical network model is necessary to realize a profitable IPTV system. Three different pricing models are analyzed and simulated in section 3. Section 4 provides a conclusion and potential future work.

2. NETWORK MODELS

This section contrasts two network models that can be used in studying the performance of P2P video content delivery.

2.1 Cloud Model

Research in P2P streaming typically considers Internet at a logical level [12][18]; it represents the Internet at large as an abstract cloud and only considers the capacity of the content server and the characteristics of the access links to related hosts. We refer this view of the Internet as the “cloud model” as shown in Figure 1.

2.2 Physical Model

In contrast to the cloud model, the physical model considers the network architecture and bandwidth constraints of the underlying links and network devices. In [9], we described and analyzed the physical model of FTTN access networks for IPTV services. The model and analysis can also be applied to xDSL or Cable connections.

As shown in Figure 2, video streaming servers are organized in two levels - a local video hub office (VHO), which consists of a cluster of streaming servers or proxies to serve viewers directly, and national super head end (SHE) offices, which can distribute videos to local serving offices based on existing policies or on demand. We concentrate on video on demand (VOD) in this paper. Each local VHO office (often referred to as “local office” below) connects to a set of access switches such as xDSL, FTTN or Cable switches through optical fiber cables. Each switch connects a community of IPTV service customers through twisted-pair copper wires, fibers or coaxial cables. A community consists of all homes which are connected to the same access (xDSL or Cable) switch. A local VHO also includes a service router to connect to a national SHE office. These uplinks (or “north-bound links”) of local offices are implemented over high-speed optical fiber networks.

The following parameters are used throughout the paper:

- \(B_{D(0)}\): Download bandwidth into a home.
- \(B_{U(0)}\): Upload bandwidth out of a home.
- \(B_{S(1)}\): Total capacity of south-bound links (downlinks) of a local access switch.
- \(B_{N(1)}\): Capacity of the north-bound link (uplink) of an access switch determined by the total bandwidth of north-bound fibers from a switch to a local VHO and the switching capacity of the service router in the VHO.
- \(u\): Average streaming bit rate for a video.
- \(N\): Maximum number of concurrent viewers supported by a local VHO.

As an example, AT&T LightSpeed network allocates 20 to 25Mbps download bandwidth (\(B_{D(0)} \leq 25\)Mbps) and 1Mbps upload bandwidth (\(B_{U(0)} \leq 1\)Mbps) to each home. LightSpeed uses an FTTN switch which has a maximum of 24Gbps downlink (or “south-side”) switching capacity (\(B_{S(1)} \leq 24\)Gbps). Each FTTN switch can connect an OC-24 fiber to a service router in a local VHO (\(B_{N(1)} \leq 1.244\)Gbps). The service router in a local VHO could then connect an OC-192 fiber to national SHE offices. Each high-definition (HD) channel uses 6Mbps bandwidth and each standard-definition (SD) channel uses 2Mbps bandwidth.
2.3 Network Constraints under Physical Model

In a physical network environment, all P2P upload traffic has to traverse through the access switches and service routers that connect the peers. As a result, P2P streaming will increase the load of access switches, local offices and national offices.

Compared with the conventional IPTV services, P2P sharing within a community may not be beneficial if the south-bound link bandwidth of an access switch is the bottleneck. However, P2P sharing within a community decreases the load on the north-bound link of an access switch. Therefore, P2P sharing within a community will have the most benefit if the infrastructure bottleneck is on the north-bound link bandwidth of an access switch.

Similarly, P2P sharing among peers across communities increases the traffic on both the north-bound links and the south-bound links of access switches. If the network bottleneck is in either \( B_{1N} \) or \( B_{1S} \), P2P sharing among peers in all communities creates more congestion for the switches and decreases the number of concurrent viewers which can be served by a local office. In this case, P2P sharing across communities is not beneficial for IPTV service providers. Also, if an IPTV service provider can apply content distribution network (CDN) technologies such as caching and replication to reduce the workload in SHE, the benefit of P2P sharing across communities in a VHO is very limited.

The detailed analysis of network constraints for P2P IPTV services can be found in [9].

3. NETWORK AT THE PHYSICAL LEVEL

A key insight of this paper is that using the “cloud model” for P2P streaming is over simplistic and misleading. More reliable results can be obtained by considering the network at the physical infrastructure level. To demonstrate our point, consider the following simple P2P algorithm. The content server receives a request for a video, identifies candidate peers with that video and spare upload capacity, and selects a random set among them to collectively serve the video. If not enough candidates are available to serve the video at its encoding rate, the server tries to serve the remaining portion itself, or denies the request if it cannot.

We simulated the performance of the system under the two models. For the physical model, we used a slice of the infrastructure of Figure 2 corresponding to one local office with 20 communities and considered the situation where the content server in the local office distributes video content to the viewers in these communities. For the cloud model, we assume the same content server and viewers are connected via the Internet cloud. We assume the same behavior for every node in the community: an idle user (i.e., the user not viewing a stream already) requests a stream with probability of 2% every time tick. A time tick occurs every minute. A peer may download only one stream at a time. There are 1000 video programs available for viewing. When a peer issues a request, it selects a program according to Zipf’s popularity distribution. Each stream lasts 120 minutes and has a data rate of 6Mbps.\(^1\) Once downloaded, the program remains available at the peer for a period called the stream time-to-live (stream TTL) with a default value of 1000 minutes. A peer may be turned off and on by its user. An operational peer is turned off with probability 0.1% on every time tick, and a non-operational peer is turned on with probability 0.5% on every tick. This means that on average every peer stays on five times longer than it stays off. We further assume that \( B_{1N} = 0.622 \text{ G (OC-12), and } B_{1S} = 10 \text{ G. Each data point in the graphs throughout the paper is obtained by running the simulation program over 5000 time clicks and taking the average over the last 2500 time ticks (when the system reached a steady state in all the simulations).}

The results for the cloud and physical models are shown in Figure 2. The figure also includes curves for the system that does not use P2P delivery under the physical model. Figure 2a shows the average number of concurrent viewers the system can support as the number of peers grows for fixed network and server capacities. The cloud model indicates that P2P delivery allows the system to serve more concurrent viewers and to scale to the growing number of viewers. However, the result is drastically different when the limitations of the physical infrastructure are brought into the picture. In fact, the cloud model could overestimate the benefit by a factor of 2 when there are more than 800 peers in a community as shown in Figure 3a. Not only does the P2P system serve fewer users, it does not scale with a growing number of users and has only a slight capacity advantage over the much simpler centralized delivery (which in fact turns to slight disadvantage for other parameter settings as seen in Figures 3b and 3c). The reason behind this drastic change is the limitations of \( B_{1N} \), the links between the local office and individual access switches. When P2P delivery occurs across different communities, two of these links are traversed: one upstream from the serving peer to the local office, and the other downstream from the local office to the receiving peer. Overall, these links are more heavily utilized under P2P delivery and more requests are denied.

Now consider the number of concurrent viewers under varying capacity of the office-to-access-switch link (Figure 2b), when the community size is fixed at 500 viewers. The results for the cloud model are not affected by this link since the model does not consider it. However, the physical model reveals an important trend: the centralized delivery becomes quickly bottlenecked at the server and stops responding to the growing bandwidth of the office-to-access-switch link. On the other hand, with P2P delivery, improvement in this link’s capacity produces a roughly linear growth in the number of concurrent viewers served, at least within the bandwidth range studied.

More differences are seen when we increase the server capacity instead (Figure 3c). In this case, the cloud model quickly reaches the point where it serves all requested streams and stops being affected by the increase in server capacity. In particular, this result might indicate that it is highly beneficial to increase the server capacity from 10 Gbps to 20 Gbps. Under physical model, however, the number of concurrent viewers is unaffected by this change. Thus, the above investment would be useless under the simple algorithm we are considering. Comparing the P2P and centralized delivery under the physical model, the centralized delivery benefits from increased server capacity until it reaches 20 Gbps, after which the bottleneck shifts to the office-to-access-switch link. However, this bottleneck transpires later than in the P2P case. Overall, Figure 3 shows that de-

\(^1\)The HD stream encoding rate is constantly improving and we expect it to reach 6Mbps soon.
pending on whether or not the network operator plans to use P2P delivery, they should focus their investment on the office-to-access-switch link bandwidth or spread it between both server and office-to-access-switch link capacities. These trade-offs cannot be revealed under the conventional cloud model.

4. COST-BENEFIT ANALYSIS

In order to encourage viewers to make their set-top boxes available for P2P sharing, some incentive may be given to peers who upload videos to other peers. This section analyzes the cost and benefit of deploying P2P technology on a physical network and compares its maximum possible profit to that of a conventional IPTV service.

4.1 Maximum Benefit for Conventional IPTV

Let \( r \) be the fee paid by a viewer in a time unit (e.g., hours or days) for video streaming services. For conventional IPTV services, the maximum revenue in a local office per time unit is

\[
R_{\text{max}} = rN
\]

where \( N \) represents the total number of viewers supported by a local office - with or without P2P incentives.

The maximum profit per time unit, \( P_{\text{nopP}} \), is

\[
P_{\text{nopP}} = \text{maximum income} - \text{IPTV expenses} = rN - E_{\text{nopP}}
\]

where \( E_{\text{nopP}} \) is the capital and operation expenses of the IPTV services per time unit.

4.2 P2P Incentive Models

To encourage P2P sharing among viewers, we consider three incentive models: Built-in model, Flat-reward model and Usage-based model.

4.2.1 Built-in Model

In this model, every set-top box includes P2P streaming software by default. Hence, P2P sharing is hidden from the viewers. The maximum profit per time unit is

\[
P_{\text{b}} = \tau N - E_{\text{pP}}
\]

where \( E_{\text{pP}} \) is the total operation and capital expenses per time unit for providing P2P IPTV services. It should be greater than \( E_{\text{nopP}} \) because P2P software needs to be installed on servers and clients and hence will increase the cost of the infrastructure. Let’s assume

\[
E_{\text{pP}} = E_{\text{nopP}} + A_{\text{pP}}
\]

where \( A_{\text{pP}} \) includes the additional software license and maintenance fees paid for P2P software and of additional hardware (such as disk storage). In the built-in model, we assume that the recurring software licence and maintenance fees and the amortization of additional hardware results in each set-top box costing \( t \) dollars extra per time unit. Therefore, \( A_{\text{pP}} = tN \). Then,

\[
P_{\text{b}} = \tau N - E_{\text{nopP}} - tN
\]
4.2.2 Flat-reward Model

In this model, a viewer signs up for the video sharing feature for a flat reward. Assume $w\%$ of viewers in a community sign up for video sharing and the reward is $d$ dollars per time unit. The total number of viewers supported by a local office is denoted to be $N$ as before. The maximum cost of incentive per time unit for the office is $dwN$. Each peer who signs up for the sharing needs to install and activate the P2P software on her set-top box. We assume that a service operator incurs the P2P software license fee only for the set-top boxes which activate the P2P software. Therefore, $E_{p2p}$ equals $E_{nop2p} + twN$. The maximum profit per time unit in this model is

$$P_f = \text{total income} - \text{expenses} - \text{incentive} = rN - E_{p2p} - dwN = rN - E_{nop2p} - twN - dwN$$

In general, $w$ depends on $d$: increasing $d$ will increase the percentage of viewers willing to share videos and hence increase $w$.

4.2.3 Usage-based model

In this model, a viewer who signed up for P2P sharing will get credit based on the number of bytes uploaded from its set-top box. Let $q$ be the credit per bit uploaded from a viewer’s set-top box and $T$ be the length of a time unit in seconds. The number of bits uploaded from peers for P2P IPTV services in $T$ seconds is $qTuN$, where $bN$ is the number of viewers downloading videos from peers among all $N$ viewers in a local office and $u$ is the average video streaming rate. The IPTV service provider gives incentives to these supporting peers based on their contributed bandwidth. In this model, the total reward given by an IPTV service provider to peers in a local office per time unit is $qTuN$. The maximum income per time unit in this model is

$$P_s = rN - E_{p2p} - qbuTN = rN - E_{nop2p} - tN - qbuTN$$

As an example to compare the maximum profit per time unit under the conventional, no-P2P model and the three incentive models, we assume that each viewer pays 3 dollars to watch a movie ($r=3$) and each movie lasts about two hours ($T=7200$ seconds). With download bandwidth $B_{0D}$ of 22Mbps, upload bandwidth $B_{0U}$ of 1Mbps, and HDTV streaming rate $u$ of 6Mbps, each HD movie consumes 43.2Gb or 5.4GB and will require six streams from peer nodes for P2P delivery. We further assume that the capital/software/operational cost of each office is $100$ million per year and the additional cost of incorporating P2P software and hardware (disk storage) on each set-top box per time unit is 10 cents. We assume that $B_{2S} = 50$Gbps. Note that $B_{2S}$ is also constrained by the total streaming throughput from the server, which is about 10Gbps.

We can now plot the profit per unit time for the conventional model vs. various incentive models of VOD services with varying $B_{1S}$ (1-30Gbps) and $B_{1N}$ (1-10Gbps) capacities, as shown in Figure 4. The maximum number of concurrent users are estimated according to a linear optimization program as discussed in [9]. In Figure 4, upper bounds for $N$ are used to illustrate the profit capacity surfaces. Typical values of $w = 0.5$, $t = 0.1$, $q = 0.01$ per Gb, and $d = 0.02$ were used to estimate these capacities. The profit number ramps up faster for the built-in model (with a given $B_{1N}$) compared to the no-P2P model as we increase the bandwidth of $B_{1S}$ until it reaches a plateau. Such analysis helps us identify where the focus of the investments should be in increasing the overall profits. Given a fixed $B_{1S}$ beyond certain capacity, ramping up the capacity of $B_{1N}$ appears to add profit to the no-P2P model linearly since it allows more users in each community without relying on the P2P technology. Note that substantial investment may be needed to increase the $B_{1N}$ capacity, which is not reflected in this graph.

4.3 Maximizing Profit Using MediaGrid Algorithm

MediaGrid algorithm [9] is a P2P sharing algorithm which selects peers for streaming or download based on physical network conditions. To study the benefit of P2P technology for an IPTV service provider under various incentive models, we performed an event-driven simulation study using the MediaGrid algorithm, with an additional enhancement allowing the stream delivery to be split between the office and peers (the original algorithm allowed split delivery only in the aftermath of peer failures). Based on the analysis in section 2, which shows that the benefit of P2P sharing among peers in different communities is very limited [9], we only consider P2P sharing within a community and simulate a system comprised of the local office and one community. We use two variations of the MediaGrid P2P sharing algo-
rithm for the simulation:
- The “peer-first” MediaGrid algorithm where peers are selected whenever the requested video can be served by peers.
- The “server-first” MediaGrid algorithm where peers are selected only when the VOD server in a VHO is overloaded.

We assume the same simulation model as described in section 2.3, using the physical network model. We assume that viewing each movie costs $3 (even if it is viewed from the local cache), peer incentive in the flat-reward model is 2.5 cents per time unit (120 minutes), and peer incentive in the usage-based model is 1 cent per upload minute. Figure 5 shows the profit numbers under the conventional no-P2P model and the three incentive models for different community sizes. As the number of peers increases, all P2P incentive models clearly generate more profit than the no-P2P model, because of the increased system capacity due to P2P content delivery. However, we see large differences among the incentive models. In fact, the usage-based model outperforms the no-P2P model for small communities because it may utilize (hence, compensate) peers even when the server has spare capacity (even in the server-first algorithm, once a peer starts uploading the stream, it continues to do so for the duration of the movie regardless of the server load). In the usage-based model, the server-first strategy generates more profits for small communities since it avoids making incentive payments to peers whenever possible. As the number of users increases, the server becomes fully utilized in both approaches and their profits converge. Finally, the built-in model always generates more profits than the other incentive models. The reason is that at the request rate used in this experiment (equal to 0.02 or one request every 50 minutes from idle viewers), the system is sufficiently utilized for the built-in model to amortize its investment on the additional hardware and software.

What happens when the system is under-utilized? Figure 6 shows the effect of lower request rates on profit, for a fixed community size of 200 viewers. As Figure 6 reveals, when the request rate is low, no-P2P model is actually slightly more profitable than all P2P models except for the usage-based model with the server-first strategy. The latter stays competitive since very few payments have to be made. Once the request rate picks up, the flat-reward model and the built-in model become more profitable since they enjoy the P2P benefits without making additional payments, while the usage-based models fall behind. In fact, the server-first profits decrease to converge with the peer-first model. This is due to the fact, that as the server utilization increases, the server-first strategy serves additional movies from the peers, and peer incentives for these movies (which can reach $7.20 for a fully peer-delivered movie under our paramters) can exceed the viewing revenue ($3).

5. CONCLUSIONS

This paper studied the conditions under which P2P technology may be beneficial in providing IPTV services. We show that the cloud model may drastically overstate the benefits of P2P video content delivery. Thus, one must consider physical network infrastructure to obtain more reliable results. Finally, we provide a cost-benefit analysis for different pricing/incentive models. In summary, P2P may not be beneficial for IPTV services unless we employ properly engineered algorithms and incentive strategies as discussed in this paper.

6. REFERENCES

Performance evaluation of P2P VoIP applications

Rodrigo Barbosa  
Universidade Federal de Pernambuco  
Recife, PE, Brazil  
rodrigo@gprt.ufpe.br

Carlos Kamienski  
Universidade Federal do ABC  
Santo André, SP, Brazil  
cak@gprt.ufpe.br

Dênio Mariz  
Centro Federal de Educação Tecnológica da Paraíba  
João Pessoa, PB, Brazil  
denio@gprt.ufpe.br

Arthur Callado  
Universidade Federal de Pernambuco  
Recife, PE, Brazil  
arthur@gprt.ufpe.br

Stênio Fernandes  
Centro Federal de Educação Tecnológica de Alagoas  
Maceió, AL, Brazil  
stenio@gprt.ufpe.br

Djamel Sadok  
Universidade Federal de Pernambuco  
Recife, PE, Brazil  
jamel@gprt.ufpe.br

ABSTRACT

In this work we evaluate the performance and behavior of two widely spread VoIP applications, namely Skype and Google Talk under different network conditions. Using a controlled environment we adopt different values for the capacities of critical links, delay, packet loss and jitter and assume the quality of received audio as the measurement of interest for evaluating its performance. We use the PESQ – an ITU algorithm that compares the original and degraded audio – in order to infer voice quality and evaluate the impact of each network parameter over the quality of received audio. Instead of ranking VoIP P2P applications, this work aims at analyzing various performance aspects and pointing out the observed weaknesses and strengths.

Categories and Subject Descriptors
C.2.5 [Local and Wide-Area Networks]: Internet; C.4 [Performance of Systems]: Performance attributes, Measurement technique; H.4.3 [Information Systems Applications]: Communications Applications

General Terms
Measurement, Performance, Experimentation.

Keywords
Peer-to-Peer Systems, Voice over IP

1. INTRODUCTION

The dissemination of Voice over IP (VoIP) technologies is considered the main enabler of telephony cost reduction nowadays. The convergence of voice and data networks makes room for a number of innovations that may change the manner people see communications. The most successful VoIP applications among end-users are those that allow free calls directly among Internet users, in a peer-to-peer (P2P) fashion, such as Skype and GTalk. Although based on the best effort Internet, a network without quality of service (QoS) assurances, such applications often achieve adequate results considering the cost-benefit and the quality standards of traditional telephony networks. These good results are the main reason for the great dissemination of VoIP applications.

One of the main factors influencing the quality achieved in a VoIP call (a session) is the voice coder/decoder (codec) utilized and its configuration parameters. However, the adoption of a codec or the selection of its specific parameters (e.g., bit rate) is not a static choice for an application. The application needs to dynamically adapt to network conditions by choosing different codecs and/or their adequate parameterization. Voice-carrying packets should not excessively be dropped, delayed or subject to high variation in delay to ensure an intelligible audio reception. Therefore, the absence of guarantees and the known variability at several time scales in the Internet traffic profile are elements that greatly impact such applications’ performance. These impairments will eventually induce its developers to deploy robust approaches to deal with a variety of traffic profiles.

This work investigates, evaluates and compares Skype and GTalk, both being free VoIP applications broadly utilized in the Internet. Considering that the codecs and their parameters are components defined exclusively by the applications and that cannot be directly manipulated, this work evaluates the behavior of the P2P VoIP applications when submitted to a number of varying network conditions. Among other contributions, this paper evaluates how applications dynamically adapt to network conditions, changing the voice flow characteristics when available capacity increases or decreases. It also evaluates the maximum delay and jitter levels accepted by voice applications for providing a satisfactory voice session and how sensitive are these applications to packet losses in the underlying network. To achieve this purpose, this work proposes a methodology involving a controlled environment for emulating Internet behavior and performing several measurements.

In the rest of the paper, section 2 presents some VoIP fundamentals; section 3 discusses important related work. Section 4 proposes the methodology and also the metrics of interest for evaluating Skype and GTalk. Section 5 presents the performance evaluation regarding the voice quality metrics and discusses their adaptation mechanisms when submitted to different network conditions. Section 6 summarizes some lessons learned and draws some recommendations for application developers. Finally, section 7 presents the conclusions.
2. P2P VoIP APPLICATIONS
The essential idea of VoIP technology is to transport voice using an IP network, conceived for transporting data packets. One of the challenges for this convergence is that the best effort Internet does not offer the quality of service guarantees that a conventional telephony network does. Therefore, a high delay in transmission, a high delay variation, or a high packet loss rate has a major impact on the quality of a voice session transported over IP [10].

Skype and GTalk allow programmers to create applications that work together with their code through a closed-source API. They use either TCP or UDP as transport protocol and both applications use the Global IP Sound (GIPS) [1] codec suite. They also inherit features from the peer-to-peer architecture, which is characterized by cooperating and sharing resources among network participants, even if some machines are hidden by NATs and firewalls. This characteristic makes them extremely robust and fault-tolerant, diminishing the possibility of service interruption. Skype relies on an overlay network with only two types of nodes: ordinary nodes and supernodes. The ordinary node is a Skype application that only performs trivial tasks, such as making/receiving voice calls. The supernodes are special nodes spread all over the Internet, besides performing the same tasks an ordinary node does, they also help the Skype network, by managing contact lists and relaying data flows when necessary. Any host with an active Skype client that is capable of receiving connections from the Internet, has a fast network access speed, has enough memory and has a fast enough processor is a candidate for being a supernode. Supernode activity is transparent to the user. There is no user choice on whether to become a supernode or not. Another P2P characteristic in Skype is the use of the hole punching technique for traversing NAT boxes.

GTalk adopts an IETF standard as a protocol, freeing its users to use other applications to communicate with. GTalk network provides interoperability with other VoIP networks and other instant messaging networks (e.g., the Gizmo Project). The service is hosted in the google.com site and can be accessed in the port 5222. We did not find any record of a GTalk overlay network.

3. RELATED WORK
VoIP applications in the Internet have attracted research on QoS for IP voice services. In [12], Shen evaluated the performance of VoIP codecs on GPRS networks and showed that the VoIP approach may create some capacity gain over traditional circuit switching, with acceptable guarantees in quality of service. Furuya [4] evaluates the relationship between network parameters (e.g., capacity and delay) and the quality of VoIP services. Although the objectives and the test environments are similar to ours, this work evaluates the dynamic behavior of popular P2P applications, while Furuya’s experiments were conducted specifically with the G.711 codec. James et al. [8] evaluate the effect of loss, delay and error recovery, among other factors, in the perceived voice quality using many codecs (e.g., G.711, G.728 and G.729).

Due to its success, a number of research studies have already been developed around Skype. Baset and Schulzrinne [1] were the pioneers in analyzing Skype (version 1.4). Their paper contains an incipient discussion of Skype’s network behavior (e.g., quantity of messages exchanged, supernodes location) in the login process, NAT and firewall traversal, call establishment and media transfer. In [11], Guha et al. carried out an experimental study between Sep/2005 and Jan/2006, focusing on the behavior of supernodes and ordinary nodes, by taking into account their exchanged traffic, life cycle and supernodes geographic location. His work serves as a base for P2P VoIP traffic modeling projects. The work of Chen et al. [9] correlates the duration of Skype calls with QoS factors: transmitted rate, delay, jitter and packet loss. Assuming that call duration may affect quality, the work defines and validates the User Satisfaction Index (USI), an index to measure user satisfaction based on QoS factors.

In [13], Suh et al. characterize Skype sessions passing through relays and propose a method to identify this type of traffic. Our work is similar to the work of Hofeld [14] in many aspects (methodology and metrics), but his experiments are restricted to 3G UMTS systems and both papers only analyze Skype.

The experiments with P2P VoIP applications conducted in this paper demand additional efforts in understanding the control policies used for application adaptation to changes in the state of the network, since the developers of the applications do not publish the algorithms responsible for such adaptation. To the best of our knowledge, this is the first research study that provides a reasonable comparison between Skype and GTalk audio quality under several network conditions.

4. EVALUATION AND METHODOLOGY
This work has the intention to analyze and to compare Skype and GTalk when submitted to adverse and favorable network conditions, under the aspects of voice quality and adaptability. We understand adaptability as the applications’ capacity and efficiency for reacting to changes in network behavior.

One way to analyze these two criteria would be through an analysis of the packet payload generated by the applications, and by doing so, discover the codec used and its parameters. Based on this information, one could infer the voice quality and adaptability through techniques previously discussed, such as Furuya [4] and James [8], or through a model to measure the quality of voice, such as the E-Model [7], a method that obtains voice quality objectively and provides the results based don factors that has influence on the audio quality (e.g., transmission delay, echo and distortions introduced by the codecs).

However, besides using a proprietary protocol, Skype communication sessions are encrypted, which prevents the analysis of packet content. Although it is possible to get essential information for the analysis of voice quality (codec and its parameters) through Skype’s and GTalk’s programming API, codecs of both applications studied are proprietary, there are no established models that allow relating these codecs to voice quality levels.

Due to these factors, the proposed methodology considers Skype and GTalk as black boxes and performs measurements at the entry and exit points of the applications (the network interface of the sender and the soundcard of the receiver) to infer performance parameters.

4.1 Experiment Environment
A testbed was built to allow the automation of the experiments (see Figure 1). Machine S (Sender) is responsible for executing the VoIP application, establishing a call and sending the audio flow to machine R (Receiver). R is responsible for executing the VoIP application, receiving and recording the audio flow from S. The software utilized for audio recording was Audacity1. The traffic from S to R was captured at both network interfaces of S and R.

1 http://audacity.sourceforge.net
NAT-S and NAT-R are NAT boxes used to reproduce the same conditions the applications face on the Internet. The network emulator, namely NIST.Net [2], emulates network conditions according to specific parameters for each experiment. Only the traffic from/to S or R to/from the Internet does not suffer interference, thus not interfering in the communication between the application and any other support peer (e.g. supernodes).

**Figure 1 – The testbed for the experiments**

We choose a network emulator instead of a simulator or real measurements in the Internet because it allows greater control over the environment and allows the replication of the experiments with the same environmental conditions. NIST.net adjusts the traffic that passes through its interfaces, while being able to modify several network parameters (unidirectional delay, packet loss rate, capacity, router queue size, etc) for different flows and representing the behavior of an entire network. Although not allowing the direct configuration of jitter, it can be approximated in the experiments through a parameter called delaySigma, which represents the standard deviation of delay.

The audio output of a CD player was connected to the microphone input in machine S while the CD player repeatedly reproduced a one-hour long audio sample, which was sent to R through the network emulator. Following recommendations by the standard used for voice quality measure, the audio consisted of a normal conversation between two people and was divided in four 15-minute parts, being two blocks of male voices and two of female voices.

It was necessary to separate the machines S and R in distinct networks because it was detected that, when both peers are in the same LAN, GTalk uses TCP for the voice sessions, while uses UDP when users are located in different LANs. Since this research aims at understanding the behavior of the applications in the Internet, it was necessary to elaborate a special network topology, as shown in Figure 1.

Both applications operate differently when the call is initiated already under adverse network conditions. On such occasions, the traffic relaying occurs frequently, and even after a readjustment for favorable network conditions, the traffic does not flow directly between the two hosts again. Since traffic relaying eliminates the controlled characteristic of the experiments, we prevent it, establishing the desired network conditions in the emulator only after the call is established.

### 4.2 Metrics of Interest

The performance parameters of the applications evaluated in this work are: a) the quality of the audio received; and b) the transmission rate from the sender to the receiver, which serves directly as a measurement for adaptability. The calculation of the transmitted rate is based on the traffic captured at the network interface of the sender. For that, the tcprat tool was used.

Although the experiments were conducted in a controlled environment, in this case it is not always possible to control all network variables. For example, when configuring a network path with a lower capacity than the required by applications, some packets are buffered and, consequently, some jitter occurs. Besides, some packets may be lost, and the received rate may be different from the transmitted rate. Knowing that packet loss and jitter do affect the received voice quality, we also measured these parameters to better understand the PES MOS results. The jitter calculation followed the method proposed in [3].

A widely adopted metric for quality evaluation of phone calls is the Mean Opinion Score (MOS) [5] standardized by ITU-T. MOS is a subjective evaluation, calculated by averaging the grades given by a large sum of people that listen to an audio sample that went through a coding/decoding process. The grade is in the range from 1 (bad) to 5 (excellent). Although its result is significant, the difficulties in performing such a large scale evaluation motivated the development of objective techniques for MOS calculation.

The E-Model [7] is a method that calculated the voice quality objectively and provides the results base don factors that influence the audio quality (e.g., transmission delay, echo and distortions introduced by the codecs). To be able to utilize the E-Model, some information relating to the functioning of the codec are necessary. Since the details of the codecs used in the GIPS (used by GTalk and Skype) are not publicly available, this work could not utilize the E-Model for voice quality evaluation. The Perceptual Evaluation of Speech Quality (PESQ) [6] estimates the MOS of a communication based on the comparison of the audio sent with the audio received. This work utilizes the PESQ MOS as a metric of voice quality.

### 4.3 Experiments Description

To evaluate applications when submitted to diverse network conditions, we created four scenarios where the following network parameters were tuned in a controlled fashion: a) network capacity in the path from sender to receiver; b) network delay; c) packet loss rate; and d) network jitter. The values of the parameters were defined based on considerations about acceptable configurations for VoIP services available in the literature [10].

The parameters were modified from a favorable to an adverse situation (aggravation) or from an adverse to a favorable situation (progression), hence generating two evaluations. Due to lack of space, only the aggravation evaluation is shown in this work.

Each value assumed for a network parameter is called a level. Levels are adjusted dynamically during a call and, for each level, the same 1-hour long audio is transmitted and 60 1-minute long samples are collected at the receiver. The experiments are repeated with the same input for both Skype and GTalk.

A preliminary evaluation was performed to discover which codecs can be used by both applications and whether they were changed.

---

2 http://www.frenchfries.net/paul/tcpstat/
during adaptation. Skype allows the monitoring of a voice session’s technical information, and among this, the codec utilized, through a simple choice selection in the application settings. In GTalk, based on the knowledge of its protocol behavior, this investigation was possible through the inspection of the packets’ payload during call signaling and during the call itself.

We carried out our experimental study between Aug/2006 and Oct/2006 using Skype version 2.0.0.81 and GTalk version 1.0.0.92.

5. PERFORMANCE EVALUATION
In this work, transmission rates are calculated including IP headers. The results shown in the graphics are the average of the samples. Vertical bars (visible only when significant) represent the confidence interval at a confidence level of 95%.

5.1 Capacity Impact
This scenario investigates the behavior of the applications when the network has critical links of varying capacities: 50, 40, 30, 20 and 15Kbps. Other parameters took the following values: 25ms delay, no explicit packet loss (only loss caused by full router queues) and no induced jitter.

![Figure 2 – Throughput for capacity variation](image)

Observing the transmitted rate (Figure 2) and PESQ MOS results (Figure 3), it is possible to observe that when the capacity of the network path is 50Kbps GTalk utilizes more bandwidth, transmitting at a rate higher than Skype, but obtains the same audio quality (the confidence intervals overlap).

![Figure 3 – PESQ MOS for capacity variation](image)

Adjusting the bottleneck link capacity to 40Kbps, GTalk clearly adapted, reducing the transmitted rate to 35Kbps. Such behavior improved its PESQ MOS score when compared to Skype, which in turn did not adapt its sending rate and continued to transmit near to the capacity limit. Both applications achieve similar results when the capacity was adjusted to 30Kbps.

With the capacity in 20Kbps, for the first time, Skype performed better than GTalk and with the capacity in 15Kbps, GTalk adapted its rate to 17.29Kbps, which was not enough to overcome Skype, which achieved a PESQ MOS 5.5% higher.

![Figure 4 – Average jitter for capacity variation](image)

An interesting observation is that despite both applications transmitted at the same rate when the capacity of the network path was configured to 20Kbps (see Figure 2), they achieved different PESQ MOS scores, as is shown in Figure 3. An explanation for this phenomenon is the occurrence of the high jitter for GTalk, as can be observed in Figure 4, which depicts the average jitter measured for both applications. Also, analysing the time series of the transmitted rate for GTalk and Skype (Figure 5), one can notice higher variability for the GTalk traffic, which causes queueing at transit network and a subsequent jitter increase.

![Figure 5 – Transmission rate for a bottleneck link of 20Kbps](image)

5.2 Delay Impact
We are now interested in the impact of the network delay. The capacity was fixed at 50Kbps (superior to the maximum transmitted rate of both applications), the packet loss was configured to be 0% and jitter for 0ms. Following Miras [10], we set the delay level to take
acceptable (1ms, 10ms, 100ms) and unacceptable values (500ms and 1000ms) for VoIP applications.

Skype's adaptation policies are clearly more sensitive to delay than GTalk's. When changing from 100ms to 500ms delay, as shown in Figure 6, the transmitted rate changed from 37.5Kbps to 19.36Kbps. However, we did not find a reasonable explanation for such behavior. GTalk did not show any sign of adaptation, since its transmitting rate did not change in this scenario whatsoever.

Observing Figure 7, Skype had a superior performance in terms of ideal network conditions, which means delays of 1ms and 10ms, while GTalk was superior with 500ms and 1000ms for delay.

5.3 Packet Loss Impact

In order to study application behavior under different packet loss rate, the bottlenecked link capacity was at 50Kbps, delay at 25ms and jitter at 0ms, while loss rates were 0%, 1%, 5%, 10%, 20%, 30% and 40%.

We raised the hypothesis that Skype added redundant information to the audio flow to reduce the impact of loss on the audio quality, which explains the increase in transmitted rate when the loss rates were 1%, 5% and 10% (disregarding IP headers, the payload was doubled in some cases).

However, these adaptations were not enough to keep the MOS value high (Figure 9). Notably, with the loss rate at 5%, Skype used more redundancy than was necessary, consuming network resources in excess without benefiting significantly its MOS score. We concluded it by observing that with the loss rate at 10%, the MOS value didn't change.

5.4 Jitter Impact

To study the impact of jitter, the capacity was set to 50Kbps, the loss to 0% and the delay to 100ms. Jitter took the levels 0ms, 20ms, 40ms, 60ms and 80ms, but measurements did not show any indication of adaptation within the applications. It is possible that in case of jitter the applications only alter the size of their buffer, without any visible impact in the transmitted traffic. As expected, the PESQ MOS metric showed a negative correlation with the jitter. In all levels both applications had a very similar behavior (not shown due to lack of space).

6. LESSONS LEARNED

Although we compare two VoIP applications, the goal of our work was not to point out which one is better application. Our aim was to find out open research questions and to come up with recommendations for VoIP application developers.

The choice of the codec has a major influence in the perceived quality and this may be verified by the fact that both Skype and GTalk use proprietary GIPS codecs, even though open codecs such as iLBC exist. However, this is not enough, since applications using the same codec may present different qualities under the same network conditions. Therefore, as far as design decisions are concerned, the use of effective adaptation mechanisms under varying network conditions primarily defines the difference in the quality obtained by VoIP applications. Since varying network conditions is currently the rule for wireless users, adaptation will increasingly play an important role, as convergence goes on. A summarization of the main findings regarding adaptation is presented below.

While adaptation usually yields better results than no adaptation at all, applications should control the impetus to adapt too fast. During the adaptation period the quality becomes unstable as we observed in some preliminary experiments with GTalk. In situations where bandwidth is seriously restricted, GTalk consistently tries to transmit at a higher rate than the permitted by the network, which induces
higher jitter levels and drop the PESQ measure. Figure 9 and Figure 3 for 20 and 15 Kbps are examples of the harmful effects of the GTalk aggressiveness. On the other hand, Skype adapts to higher delays by sharply decreasing the transmission rate. While we could not find any reason for that design decision in the literature, it clearly results in worse quality levels.

Adaptations that send redundant information may yield positive results, as depicted by Figure 8 and Figure 9 in the loss scenario. Skype was able to keep PESQ MOS values above 3 up to 10% packet loss with such adaptation, while GTalk did not adapt at all and the quality decreased linearly with the increase in loss rate. However, there is a trade-off in increasing the redundancy level (and bitrate) and the improvement obtained, since transmitting at higher rates is not desirable in most voice systems. More research is needed to come up with smarter redundancy schemes targeted to specific network variation conditions (especially in wireless environments). In general, we think that codecs should be developed having in mind different adaptations strategies, i.e. they should not only be designed to allow different choices to be made, but to assume that these choices would be changed in runtime according to different network conditions.

Finally, adaptations that make triangulations with the supernodes might be performed as a last resort when the quality keeps unacceptable for longer periods. As a matter of fact, GTalk used to rely on this mechanism in previous versions, although it has not been observed in this version. However, considering that triangulation is a workaround for congested routes and frequently congestion occurs at access networks that have single paths, the real benefits of such mechanism would be an interesting research topic.

7. CONCLUSIONS AND FUTURE WORK
This work compared the performance of two P2P VoIP applications: Skype and GTalk. We discussed the dynamic adaptation policies and evaluated the audio quality of applications through the observation mainly of the PESQ MOS and the transmitted rate.

This work showed that Skype and GTalk rely on estimators to analyze the quality of the service offered by the network and control the use of their codecs. Despite both companies affirm their applications use the GIPS codec library, sometimes both the characteristics of the audio stream and the measured audio quality were different under equal scenarios.

Under ideal network conditions, it was observed that, although the difference is very small, the audio transmitted by Skype suffers less degradation than that by GTalk but, under high delay Skype performed unnecessary adaptation. We conclude that GTalk does not implement any mechanism for adaptation when submitted to packet loss, whereas there are strong indications that Skype uses a data redundancy mechanism against loss. Under high levels of jitter, none of the applications adapted their sending rate.

When considering the voice quality aspect for both applications and also including the experiments that vary from an adverse to a favorable situation (progression), one will see that Skype slightly overcame GTalk in 24 occasions, while GTalk performed better only in 4 scenarios and in 14 occasions there was a draw. However, in most occasions where Skype performed better, the PESQ MOS difference was below 0.1, showing that even with Skype’s apparent advantage, both applications are very close in terms of voice quality.

It is known that PESQ MOS is not the most adequate algorithm to evaluate voice quality for high values of delay. Therefore, despite the results involving delay variation being interesting, conclusions must be taken carefully. The study of the friendliness of both applications towards TCP flows and also between both applications when competing together for network resources is also an interesting future work.

8. REFERENCES
Measuring P2P IPTV Systems

Thomas Silverston
Université Pierre et Marie Curie - Paris 6
Laboratoire LIP6/CNRS, UMR 7606
104 avenue du président Kennedy
75016 Paris, France
thomas.silverston@lip6.fr

Olivier Fourmaux
Université Pierre et Marie Curie - Paris 6
Laboratoire LIP6/CNRS, UMR 7606
104 avenue du président Kennedy
75016 Paris, France
olivier.fourmaux@lip6.fr

ABSTRACT

P2P IPTV systems start to be very popular on the Internet. Measuring the impact they have on the network and understanding their behavior is an important field. Available applications are based on proprietary solutions, thus traffic analysis is the only feasible way to identify the mechanisms used to get video streams. In this context, during the 2006 FIFA World Cup, we performed an extensive measurement campaign. During this worldwide event, we measured network traffic generated by the most common P2P IPTV applications, namely PPLive, PPStream, SOPCast, and TVAnts. Our observations show that all these applications generate different traffic patterns and use different underlying mechanisms. Each application has its own download policy and maintains a different set of peers. From the traces we collected, we extract several statistics, which help in having a better understanding of the behavior of P2P IPTV systems.

1. INTRODUCTION

The success of P2P file sharing and VoIP applications has proved that the P2P paradigm is an efficient solution to deliver all kinds of content over the Internet. Nowadays, there are P2P video live streaming applications (P2P IPTV) that have been successfully deployed on the Internet. These applications are proprietary and claim to be swarming protocol like Donet [1]. In these P2P systems, data are divided into chunks and each peer exchanges with other peers information about the chunks they have. Then each peer is able to download data chunks from several peers concurrently. However, the exact way of working of these emerging applications is still widely unknown. In this context, it is important to evaluate the traffic impact of these P2P applications on the Internet. Even though lots of measurement studies have been conducted on P2P file sharing and telephony systems, very few tackled P2P IPTV.

In this paper, we make comparisons between different applications by analyzing the different traffic patterns we collected. Through these analyses, we highlight design similarities and differences and point out global behavior in these P2P networks.

During the 2006 FIFA World Cup, we measured network traffic generated by several P2P IPTV applications. We collected packet traces by using the following applications: PPLive [2], PPStream [3], SOPCast [4] and TVAnts [5]. We chose these applications because they are the most popular on the Internet. We focused on collecting traffic during World Cup because it is a large-scale event, which exhibits a live interest for the users. Thanks to all the collected data, we obtained a representative sample of large-scale P2P IPTV applications. Our work differs from Hei [6] by the number of applications we studied and the followed comparative approach. To the best of our knowledge, this is the first study, which focuses on a large-scale live event broadcasted on P2P networks.

Results in this study focus on a single event day where two soccer games were scheduled and we analyzed the traffic generated by the four previously mentioned P2P applications. We limit the scope of our analysis to these traces since we noticed their representativeness of our data set. Our results show that all the applications generate different traffic patterns. The measured download traffic indicates that the applications use different mechanisms to get the video and they maintain a different peers neighborhood.

The remainder of this paper is organized as follows. In Section 2 we present the related work. In Section 3 we describe our measurement experiment set-up. Then in Section 4 we present and discuss the measurement results. We conclude and expand our future work in section 5.

2. RELATED WORK

Measuring P2P live streaming systems is still an emerging topic, but there are previous measurement studies about live streaming media delivered on the Internet. Sripanidkulchai et al. [7] show that large-scale live streaming can be supported by P2P end-users applications despite the heterogeneous capacity of peers. In P2P IPTV systems, Zhang et al. [8] present their own P2P IPTV system and give network statistics like users’ behavior in the whole system and the quality of video reception. Hei et al. [6] have a similar work to ours. They study an existing P2P IPTV application by collecting packet traces. Our work is different from theirs since we do not focus on a single application but on several applications. It helps us to highlight design differences and to infer global behavior of such P2P network without being strongly related to a single P2P system implementation. We
collected a larger and more various data set from a representative panel of applications during an entire large-scale event. Finally, an important distinction between Hei works and ours come from the live interest of the measured event. It is intuitive but corroborated by Veloso et al. [9] that traffic patterns have not the same characteristics whether broadcasted content exhibits a live interest for users or not.

3. EXPERIMENT SETUP

Our measurements started with the 2006 FIFA World Cup from June 2nd to July 9th. We collected a huge amount of data, measuring most of the World Cup games with different applications at the same time, under two different network environments (campus Ethernet access and residential ADSL access). In this paper we focus on comparisons between four P2P IPTV applications according to their traffic pattern. In all our data, we selected packet traces on June 30 because they are well representative of all of them. Two soccer games were scheduled: one in the afternoon (Germany vs. Argentina) and one in the evening (Italy vs. Ukraine). Fig. 1 describes our measurement experiment platform. We used two personal computers (PCs) with 1.8 GHz CPU, and common graphic capabilities. For the rest of this paper, the PCs will be called nodes. Operating system was Windows XP because all the applications have been implemented for this OS. The two nodes were situated in our campus network and were directly connected to the Internet with 100 Mbps Ethernet access. We used tcpdump to collect the packets and their payload generated by the applications. During each game, the nodes were running tcpdump and a distinct P2P application. The Ethernet cards did not suffer any packet loss and captured all the packets. All the measurements have been done watching CCTV5, a Chinese sport channel available for all the measured applications. All the applications used MPEG4 video encoding. We did not measure the traffic between the two consecutive games. From the first game to the second one, we only changed the running P2P applications on the nodes to get packet traces from different application. At the end of the experiments, we collected four packet traces: one per application. We chose to measure the first game by running PPStream and SOPCast on the nodes, and the second one by running PPLive and TVAnts. Collected packets can only provide from a node or from remote peers in the Internet. Table 1 summarizes the four collected traces.

The two measured events are similar (a soccer game in the FIFA world cup) and exhibit the same live interest for users. We analyzed our packet traces by developing our own perl parsing tools.

3.1 Data Collection Methodology

We differentiate TCP sessions according to TCP Flags and we only take a session into consideration if at least one of its TCP segment has a payload. Session durations are driven by TCP segment payload. A session start time was calculated as soon as we received (or sent) a TCP segment with a data payload. The session duration was increased for each new TCP segment with a payload. A session ended when we received an explicit flag, but the end session time was the instant where we received the last TCP segment with payload. The session duration depends only on TCP segment with payload. We compute session duration relying on UDP in the same payload-driven way.

4. MEASUREMENT ANALYSIS

In this section, we show the results of our measurement study. We first analyze the traffic of all the applications. Then, we point out the download policies used by the applications to get the video and the peers neighborhood they maintain. Finally, we show the video peers lifetime for all the applications.

4.1 General Observations

Table 1 shows that PPStream relies only on TCP. Major part of PPLive traffic relies on TCP whereas SOPCast traffic relies mostly on UDP. TVAnts is more balanced between TCP and UDP. All the applications have download throughput quiet constant and slightly above the video bi-
Most part of TVAnts traffic is transported by TCP (signaling and video sessions both on TCP or UDP). However, we can distinguish signaling and video clusters but they pattern shows a balanced use of TCP and UDP (Fig. 3(d)). Compare to the other measured applications, TVAnts traffic video traffic on UDP. We currently have finer analysis to and on the right top (video) of the plot but they are not UDP. We can still observe clusters in the middle (signaling) and the one in the middle of the plot is for signaling sessions (small packets and short session duration) and the one in the right top of the plot is for video sessions (large packet and long session duration). PPLive and PPStream use TCP to transport video and signaling traffics. We are still investigating the difference between signaling sessions relying on UDP or TCP for PPLive. Regarding SOPCast traffic pattern (Fig. 3(c)), we observe that it uses almost exclusively UDP. We can still observe clusters in the middle (signaling) and on the right top (video) of the plot but they are not so clearly formed. SOPCast transports both signaling and video traffic on UDP. We currently have finer analysis to understand why there are very few sessions relying on TCP. Compare to the other measured applications, TVAnts traffic pattern shows a balanced use of TCP and UDP (Fig. 3(d)). We can distinguish signaling and video clusters but they both contain TCP and UDP traffic. TVAnts transports signaling and video sessions both on TCP or UDP. However, most part of TVAnts traffic is transported by TCP ($\approx \frac{75\%}{1}$, table 1). Table 2 summarizes our observations from Fig 3: all the measured applications have different traffic patterns. PPStream uses only TCP for video and signaling traffic while PPLive adds UDP for some signaling traffic whereas TVAnts uses both TCP and UDP for all kind of traffic and SOPCast uses almost entirely UDP. This is an important difference between all the applications. If we take into account all the applications, we observe that the video traffic is mostly transported by TCP, which is not a transport protocol dedicated for multimedia and real-time applications. As for common Internet video streaming applications, TCP could be justified to reach all kind of Internet users, even if there are behind filtering or NAT systems.

To evaluate the amount of signaling traffic overhead in the traces, we separated video and signaling traffic with an heuristic proposed in [6]. The heuristic works as follows: for a session (same IP addresses and ports), we counted the number of packet bigger or equal than 1000 Bytes\(^1\). If a session had at least 10 large packets, then it was labeled as a video session and we removed small packets (< 1000 Bytes) from this session. We removed all non-video sessions from the traces. Table 3 presents the results of this heuristic for the four traces. The signaling overhead ratio is different for all the traces (from 4.1% to 19.3%). SOPCast overhead is more important than the other and PPLive has the lower signaling overhead. PPStream and TVAnts have almost the same signaling overhead ratio. PPLive and PPStream have similar traffic patterns, but at the end, the signaling overhead needed to manage the P2P network is different. PPLive and PPStream should not use the same underlying mechanisms. PPStream and TVAnts have a similar overhead ratio but present different traffic patterns. Their underlying mechanisms should also be different, as SOPCast, which has the more important signaling overhead ratio and a specific traffic pattern. In the next sections, we give some insights about underlying mechanisms used by all the presented applications.

### 4.2 Traffic Patterns

These P2P applications are proprietary but claim to use swarming mechanisms where peers exchange information about data chunks and neighbor peers as in Donet ([1]). In such P2P network, a peer will iteratively discover other peers and would establish new signaling or video sessions. Video sessions are likely to have long duration because users want to watch the entire game whereas signaling sessions are likely to be shorter in time. Furthermore, video streaming packets size is expected to be large and signaling session packets size is supposed to be common. Fig. 3 plots the average packet size according to the session duration using a log-log scale. PPLive (Fig. 3(a)) and PPStream (Fig. 3(b)) have similar TCP traffic patterns but PPLive uses UDP too. PPLive UDP sessions vary from short to long duration, but their average packet size is small and constant. PPLive UDP traffic transports signaling sessions. PPLive and PPStream exhibit two clusters in their TCP traffic patterns: the one in the middle of the plot is for signaling sessions (small packets and short session duration) and the one in the right top of the plot is for video sessions (large packet and long session duration). PPLive and PPStream use TCP to transport video and signaling traffics. We are still investigating the difference between signaling sessions relying on UDP or TCP for PPLive. Regarding SOPCast traffic pattern (Fig. 3(c)), we observe that it uses almost exclusively UDP. We can still observe clusters in the middle (signaling) and on the right top (video) of the plot but they are not so clearly formed. SOPCast transports both signaling and video traffic on UDP. We currently have finer analysis to understand why there are very few sessions relying on TCP. Compare to the other measured applications, TVAnts traffic pattern shows a balanced use of TCP and UDP (Fig. 3(d)). We can distinguish signaling and video clusters but they both contain TCP and UDP traffic. TVAnts transports signaling and video sessions both on TCP or UDP. However, most part of TVAnts traffic is transported by TCP ($\approx \frac{75\%}{1}$, table 1).

Table 2 summarizes our observations from Fig 3: all the measured applications have different traffic patterns. PPStream uses only TCP for video and signaling traffic while PPLive adds UDP for some signaling traffic whereas TVAnts uses both TCP and UDP for all kind of traffic and SOPCast uses almost entirely UDP. This is an important difference between all the applications. If we take into account all the applications, we observe that the video traffic is mostly

<table>
<thead>
<tr>
<th>Signaling overhead</th>
<th>PPLive</th>
<th>PPStream</th>
<th>SOPCast</th>
<th>TVAnts</th>
</tr>
</thead>
<tbody>
<tr>
<td>ratio</td>
<td>4.1%</td>
<td>13.6%</td>
<td>19.3%</td>
<td>10.2%</td>
</tr>
</tbody>
</table>

---

\(^1\)1000 Bytes instead of the 1200 Bytes proposed by the heuristic because it fitted better our traces.
traffic and the top peer contributes to almost all the traffic during its session duration. The top peer session is quiet short regarding all the trace duration. These observations suggest that PPLive gets the video from only a few peers at the same time and switches periodically from a peer to another one. Remember PPLive and PPStream have almost the same traffic pattern (Fig. 3(a) 3(b)), it is interesting to observe that PPStream download policy is the PPLive opposite. For PPStream (Fig. 4(b)) the top ten peers do not contribute to a large part of the download traffic and neither the top peer. PPStream has to get the data from many peers at the same time and its peers have long session duration. SOPCast top ten peers (Fig. 4(c)) contribute to about half the total download traffic and top peer contributes to all the top ten peers traffic during its session duration. In a way, SOPCast download policy looks like PPLive policy: it switches periodically from provider peer. However, SOPCast seems to always need more than a peer to get the video compare to PPLive where a single peer could be the only video provider. TVAnts download policy (Fig. 4(d)) seems to mix PPStream and SOPCast policies. TVAnts top ten peers contribute to about half the total download traffic (like SOPCast), but top peer does not contribute to a large amount of the total traffic (like PPStream). TVAnts top peer does not contribute as few as PPStream’s one but does not stay as long as PPStream top peer.

If we summarize our observations, the presented applications implement different download policies and do not expect peers to have the same capabilities. Some download policies expect peers to stay in the network for a long time (like PPStream) or short time (PPLive, SOPCast), or expect a peer to have huge capacities to send all the video (PPLive) or low (PPStream, TVAnts). According to the application, a peer can get the video from only a few peers at the same time or from many peers and its session duration is various. Different download policies highlight differences to maintain a neighborhood for a peer to get the video. This will be point out in the next section.

4.4 Peers Neighborhood

In swarming P2P systems, peers have to maintain peers neighborhood to get the data chunks from several peers at the same time. In Fig. 5, we plot for each application the video download peers neighborhood maintained by our nodes during all the traces duration. A video download peer is a peer, which has sent video to our controlled nodes. In
the following, we will refer to the number of video download peers as \textit{VDP}.

PPLive maintains a relatively low and constant VDP whereas PPStream has a high and constant VDP. SOPCast’s VDP can be as high as PPStream’s one but fluctuates largely. As expected, SOPCast has no VDP when our node running SOPCast receives no traffic. TVAnts VDP number is high and also fluctuates.

All the applications maintain a different peers neighborhood, which corroborates the applications have different download policies to get the video. As expected, there is a large set of steady peers for PPStream, only a reduced set for PPLive. SOPCast and TVAnts have high and fluctuating VDP. VDP fluctuations are observed for applications, which use an important part of UDP traffic (table 1). These VDP fluctuations may come from the non reliability of UDP, which causes more packet losses and forces peers to make its VDP always evolving to get the video.

**4.5 Video Peers Lifetime**

In P2P IPTV, end-hosts are responsible to duplicate flows to each other. End-hosts are not entities dedicated to stay in the networks all time: they can join or leave the network whenever they want and are prone to suffer failures. P2P IPTV systems have to deal with the arrivals and departures of peers (churn of peer). It is a challenging issue because live video has to respect playback instant to get smooth playback quality. A high churn of peers will involve additional delays or jitter variations for packet delivery, which will decrease overall video quality. In this section, we show the video peers lifetime to point out the churn of peers. Since our nodes have only a local view of all the peers in the network, the video peer lifetime is the duration between the first time and the last time our controlled nodes exchange video traffic with another peer.

As an example, Fig. 6 plots the TVAnts complementary cumulative distribution function (CCDF) of video peers lifetime. For all the applications, the video peers lifetime CCDF follows a Weibull distribution. The CCDF plots for the other applications can be found in [10]. The Weibull distribution functions used to fit the measured video peers lifetime are presented in table 4. It also shows average peer lifetime. The Weibull distribution is an exponential-like distribution often used in reliability testing and failure analysis. For all the applications, there are no more than 10% of peers, which stay in the network during an entire game. Moreover, the
average video peers lifetime is different for all the applications and far from an entire game duration. The departure of a peer can be due to an user, which stops to watch the game or due to the application’s mechanisms, which force a peer to switch from a video peer to another one. Since all the applications exhibit a Weibull distribution for video peers lifetime, our meaning is that Weibull distributions are driven by users’ behavior while the different average video peers lifetime comes from the underlying mechanisms used by the applications.

Thanks to our measurement observations, we have a better understanding of P2P IPTV systems. This knowledge will be used in our other works to model and simulate these systems.

ACKNOWLEDGMENTS
This work is supported by the European Union under the IST Content project (FP6-2006-IST-507295).

6. REFERENCES
Challenges with developing a Commercial P2P System

Aaron Colwell
RealNetworks
Outline

• Why P2P?
• RN Content Delivery Profiles
• Design Challenges for a Commercial P2P System
• Insights from our P2P Bandwidth Savings Study
• Future Research Directions
Why P2P?

- Economical, scalable content delivery
  - Reduction in bandwidth costs and server capacity
    - BitTorrent claims of 80+% savings are hard to ignore.
  - Availability increases with popularity -> no overprovisioning for flash crowds.
    - Large-scale events require us to maintain large server farms for events that only happen a few times a year.
Content Delivery Profiles

- Have several different delivery profiles to address
  - On-demand music service (Rhapsody)
    - Millions of clips
    - Typically 3-7 minutes, ~4-6 MB each
  - Large Live Broadcast events (Real Broadcast Network)
    - Big Brother
    - Sub-10 second latency, w/ 10s of thousands of users.
  - Movie & Casual Game Downloads (Film.com, RealArcade)
    - 10-100s of MB
    - 100-1000s of titles.
Challenges for P2P networks

• NAT Traversal
• Content Integrity
• System Security
• Churn
• Fairness
• Peer Heterogeneity
• Quality of Service
• Participation Incentives
Legal vs Illegal Content Challenges

- Illegal Content
  - Best effort service acceptable.
  - Don’t expect QoS guarantees.
  - No financial investment in content.
  - Willing to take chances with potentially malicious software to gain access to the content.

- Legal Content
  - Expect a base level of QoS.
  - Content must always be available, esp. if they are paying money.
  - Customers wary of “unnecessary connections” from commercial products. Acceptance requires consumer education & incentives.
  - Content & delivery network must be secured to keep rights holders happy.
Where do we start?

• Focus on subset of the challenges that capture important aspects of our various delivery profiles.
  • Churn
  • Peer Heterogeneity
  • Quality of Service

• Determine how these challenges affect BW savings in a peer assisted CDN
P2P Bandwidth Savings Study

- Study BitTorrent to understand potential BW savings for RN workloads.
- Explore dimensions that likely affect BW savings
  - Peer BW heterogeneity
  - Arrival/Departure processes
  - Seeding Strategies
- Determine whether BW saving are worth the effort of developing a P2P delivery system.
Evaluation Environment

Multiple BT clients (peers) run across 6 hosts

Host

Link w/ varied upload/download capacity

BitTorrent client for file \( n \)

BitTorrent tracker + seed for file \( m \)
WARNING

Graphs are intended to display trends and provide intuition about behavior.

The data is preliminary and should not be taken as actual BW savings.
BW Savings - Impact of peer UL/DL ratio

- For fixed UL rate, as DL rate ↓, BW savings ↑
  - Lower DL rate → longer transfers & fewer peers to saturate link → more data from peers instead of origin

Games Workload - %BW Savings over HTTP
Max Upload Rate = 250Kbps

Music Workload - %BW Savings over HTTP
Max Upload Rate = 250Kbps

W-d x u = WAN peer dl-rate x up-rate
Impact of Content Availability at Peers*

- Content availability at peers influenced by peer seed time and file inter-reference time
  - Both factors can be captured by #Seeds:#Leechers
- BitTorrent tends to favor downloading from seeds
  - Previous results used 1 seed and n-1 leechers (i.e., ‘worst’ case)
- More seeds → better availability → more BW savings
  - Provide incentives to seed (inherent for live content)

* Max UL rate = 250 Kbps Max DL rate = 750 Kbps
“Smarter” seeding to minimize BW utilization at origin

- **SuperSeeding** mode (origin masquerades as leecher)
- Explicitly cap upload rate at origin
- Significant savings in BW w/ superseeding
- But...

---

Games Workload - %BW Savings over HTTP

<table>
<thead>
<tr>
<th>%BW Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
</tr>
<tr>
<td>70</td>
</tr>
<tr>
<td>50</td>
</tr>
<tr>
<td>30</td>
</tr>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># Clients</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>24</td>
</tr>
<tr>
<td>48</td>
</tr>
</tbody>
</table>

Music Workload - %BW Savings over HTTP

<table>
<thead>
<tr>
<th>%BW Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
</tr>
<tr>
<td>70</td>
</tr>
<tr>
<td>50</td>
</tr>
<tr>
<td>30</td>
</tr>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># Clients</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>24</td>
</tr>
<tr>
<td>48</td>
</tr>
</tbody>
</table>
Mean DL rates w/ SuperSeeding

- Mean DL rate at clients significantly lower w/ superseeding
  - Often < file encoding rate (e.g., <150 Kbps for music)
- Origin cannot attempt to reduce BW too aggressively if QoS matters
Study Conclusions

• Significant savings can be realized for a variety of workloads.
• Peer BW, mesh composition, and seeding strategy have complex interactions that vastly affect bandwidth savings.
• Key parameters need to be identified to help control BW savings.
Future Research Directions

- Further explore the parameter space covered in the study.
- Study how fairness, security, resource utilization, etc. affect BW savings.
Questions
Bringing TV to the PC

Sugih Jamin
University of Michigan and Zattoo, Inc.

NOSSDAV ‘07 June 5th, 2007 (abridged)
Live TV On the Internet!

Zattoo the Program
• Watch live TV on your computer
• Only need Internet connection
• Linux, Mac, and Windows version

Zattoo’s Benefits:
• One user interface
• Better video quality
• Significant cost reduction
350K Unique Subscribers in Switzerland

Company founded in May 2005
Launched in June 2006 with 4 channels

Unique subscribers

Source: User Database
Zattoo EXTENDS TV Reach

Swiss Zattoo subscribers
Thousands

<table>
<thead>
<tr>
<th>Month</th>
<th>Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aug</td>
<td>4</td>
</tr>
<tr>
<td>Sep</td>
<td>7</td>
</tr>
<tr>
<td>Oct</td>
<td>12</td>
</tr>
<tr>
<td>Nov</td>
<td>23</td>
</tr>
<tr>
<td>Dec</td>
<td>41</td>
</tr>
<tr>
<td>Jan</td>
<td>87</td>
</tr>
<tr>
<td>Feb</td>
<td>197</td>
</tr>
</tbody>
</table>

2006

2007

Triggering factors driving adoption
Percentage

- "I use Zattoo because…"
- "I want to watch TV while doing other stuff with my PC"
- "The TV is occupied"
- "I'm in a different place than my TV"
- "I don’t have a TV"

Other reasons

- 17%
- 22%
- 29%
- 14%
- 18%

- 350’000 users in Switzerland (10% of all broadband users) in just over 6 mos.
- Only thanks to word of mouth and media coverage
- 70% uses Zattoo at home
- Zattoo is not a substitute but an addition and an extension to classic TV viewing
- Broadcasters profit from extended reach

Source: Zattoo user database; Schober Partners Group study of Zattoo users, November 06-February 07
Industry Overview

Internet TV companies and products

- 2 h or shorter
- 24 x 7

Not live
- YouTube
- Joost

Live
- MLB TV
- JumpTV

Competitive advantages of Zattoo
- Particularly strong in live (8-15 secs lag) and 24x7 video
- Rapid (3-8 secs) channel switching
- End-user focus with user friendly interface
- Proven ability to license and broadcast high quality live content

Areas of strength for Zattoo
Current focus

Octoshape
UUSee
Etc.
Internet TV Delivery at Low Cost

Zattoo uses the computing and networking power of the end users’ computers

Key capabilities and their impact

- **Self-expanding network**
  - De-bottlenecks content source; load is smoothly distributed among viewers

- **Traversing firewalls**
  - Works for any broadband connection, even behind firewalls (campus, corporate settings, home)

- **Compensating for bandwidth asymmetry**
  - Reduces subsidy required to keep network operating, while respecting users’ low upstream capacity

- **Fault tolerance in the face of peer churn**
  - Prevents loss of service when many peers join or leave a stream at the same time

- **Smoothing out transient congestion**
  - Reduces frame drops and stalls, and improves streaming smoothness
Components of the Zattoo Network

Broadcaster:
• Hooks up a broadcast server with H.264 card to feed live TV into Zattoo
• Recommended bandwidth is 1-10mbps uplink

Backend Services:
• Channel control
• Authentication and DRM license server
• Rendezvous host
• Accounting and Billing
• Statistics collection

Zattoo Client:
• Standalone app
• Authenticates user and location
• Registers with RH upon startup
To Watch a Channel

- Per player authentication (steps 1 and 2)
- Per channel DRM checking (steps 3 and 4)
- Tuning in (step 5)
- Getting reception (step 6)
Zattoo Infrastructure

Broadcast Servers
- Satellite Dish
- Coax Multiplexer
- Servers with Signal Decoder and Capture
- Router

Backend Services
- Rendezvous host
- Authentication and DRM license server
- Webstats
- Channel Server
- Router

PC with viewer
- Supemode Client
- Zattoo's Repeater node
- Router
- PC with viewer
EXPANDING TO EUROPE, NORTH AMERICA AND ASIA

Roadmap and market potential
millions

• Europe*
  – Planned market entry:
    Q3 2006 - Q4 2007
  – Broadband connections: 59.9

• North America
  – Planned market entry:
    Q2 - Q3 2007
  – Broadband connections: 58.4

• Asia**
  – Planned market entry:
    Q4 2007 - Q1 2008
  – Broadband connections: 37.8

* EU 15
** Japan, South Korea, Australia
Source: OECD Broadband Statistics, December 2005
Servers Shipment and Setup
Peer-Assisted Delivery: The Way To Scale IPTV To The World

Jin Li, Principal Researcher
Communication and Collaboration Systems
Microsoft Research
Internet Traffic on the Rise

- Internet traffic trend: grow at a compound monthly average of 7.4% in 2006

Traffic at Amsterdam Internet Exchange (AMS-IX)
Video Fuels the Growth

- Video streams served increased 38.8% in 2006 to 24.92 billion
  (Source: AccuStream iMedia Research)
- 53 web-video startup in 2006, $521M VC funding (Source: DowJones VentureOne)
- Major studio goes into Web Video

- $410M video ads revenue in 2006 and grow by 89% in 2007
  (0.6% of $74B TV ad market, Internet ads $16.4B in 2006, expect to grow 19% in 2007)  [source: Emarketer]
Just Build More Powerful Data Center?

- VHS quality video streaming: 500 kbps (H.264/VC-1)
  - 200,000 viewers = 100 Gbps
  - DTV quality: 1Mbps
  - DVD quality: 1.5-2Mbps
  - HD quality: 6-10Mbps
- Data center capacity: Tera Grid
  - 30 petabyte of storage
  - 40 Gbps backbone: 80k video viewers
  - 70 days to retrieve all data through the backbone network
Can CDN Help?

- Akamai
  - 20,000 servers, 900 point of presence, 71 countries
  - 300Gbps bandwidth
- Limelight
  - 25 point of presence, hundreds servers per presence
  - 1,000Gbps bandwidth
- Combined: 2.6 million viewers.

- Current TV audience
  - Olympics: 2.5 billion viewers
    - Each viewer may have his/her own interest (different sport event, athlete nationality, etc.)
P2P is the Way To Scale

- Economical to run
  - Saves server/CDN bandwidth, disk I/O, CPU, memory
- Robust
  - no single point of failure in network
- Super-scalable
  - system capacity increases with number of nodes
P2P Traffic Dominates

- 1999 to present: fueled by Napster, KaZaA, eDonkey and BitTorrent

CacheLogic Research
Internet Protocol Breakdown 1993 - 2006

50-65% of downstream traffic is P2P,
75-90% of upstream traffic is P2P.
Internet Traffic Statistics

- 60+% of Internet backbone traffic is P2P
- Among P2P traffic, 65% is video, 24% is audio [source: CacheLogic]
P2P Benefit Content Owner

- Based on actual MSN video trace
  - Assume same user behavior, client-server vs. peer-assisted VoD

<table>
<thead>
<tr>
<th></th>
<th>client-server VoD</th>
<th>peer-assisted VoD</th>
<th>Peer-assisted VoD (3x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server BW</td>
<td>1230 Mbps</td>
<td>33 Mbps</td>
<td>389 Mbps</td>
</tr>
<tr>
<td>Saving</td>
<td>40x</td>
<td>3x</td>
<td></td>
</tr>
</tbody>
</table>

Savings breakdown (Apr. 2006)
P2P Benefit Consumers: Better Video Quality, More Selection
P2P Should Be ISP Friendly

P2P: 60+% of Internet backbone traffic. ISP Friendly ➔ Reduce transit traffic.

Tier-1 ISPs
AS 1 — AS 2

Tier-2 ISPs
AS 3 — AS 4 — AS 5

Tier-3 ISPs
AS 6 — AS 7 — AS 8 — AS 9 — AS 10

128.16.0.0/16

Peering/sibling relationship
Customer/provider relationship
P2P Locality: Intra ISP
IPTV Experiments and Lessons Learned

Panelist: Klara Nahrstedt
Panel: Large Scale Peer-to-Peer Streaming & IPTV Technologies
PPLive IPTV System

- Well-known IPTV system
  - 3.5 M subscribers in 2005
  - 36.9 M subscribers in 2009 predicted
  - May 2006 – over 200 distinct online channels
  - Revenues could up to $10 B
  - Need to understand current system to design better future systems

- PPLive Architecture
  - Management Server
    - Retrieve list of channels via HTTP
  - Membership Server
    - Retrieve small list of members nodes of interest via UDP
  - Other Peers
    - Learn about other partner peers by periodically probing via UDP
PPLive IPTV Measurements

- In 2006/2007 – multiple commercial P2P live systems evaluations/measurements were conducted
  - [2] X. Hei et al in WWW’06 workshop on IPTV Services

- Measurements of network-centric metrics
  - Video traffic, TCP connections

- Measurements of user-centric metrics
  - Geographic distributions, user arrival and departure, user-perceived quality

- Measurements of overlay-based characteristics
  - Size of overlays, average degree of a peer, availability relation between peers, user behavior, session length, channel population size
Methodology of Study in [3]

- Crawler-based measurement study
- UIUC machines or Planetlab machines join PPLive channel and then crawl peers that joined the channel
- Crawler collects information and we get snapshot

<table>
<thead>
<tr>
<th>Chan. size</th>
<th>PS Len</th>
<th>#Pro</th>
<th>Pro Len</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 32K-45K</td>
<td>6h15m</td>
<td>6</td>
<td>36m-2h</td>
<td>Movie</td>
</tr>
<tr>
<td>B 10K-15K</td>
<td>4d4h</td>
<td>300</td>
<td>20m</td>
<td>Cartoon</td>
</tr>
<tr>
<td>C 8K-12K</td>
<td>1d2h16m</td>
<td>40</td>
<td>40m</td>
<td>Movie</td>
</tr>
</tbody>
</table>
Findings

- Average Node degree is independent of channel population size
Findings

- Peer Availability – Bimodal Relation
  - Nodes in same snapshot have correlated availability
  - Random node pairs have independent availabilities
Findings

- PPLive peers are impatient
Findings

- Channel Population varies widely over a day
  - Channel population size variations are larger than in P2P file-sharing networks
Lessons Learned

- Future media P2P streaming systems in their churn models need to take into account the bimodal distribution of peers availability
- Homogeneous protocols and homogeneous design proved to be quite good, i.e., the PPLive protocols that treated peers equally are simpler and seem efficient.
- User interfaces for IPTV could contribute more efficiently to the P2P overlay infrastructure
- More complex IPTV behavior is possible that needs to be taken into account by the P2P streaming systems
TV Channel Selection

**Problem:**
- Enable TV channel selection based on
  - user interest
  - channel content

**Solution:**
- Proactive TV channel search
- Asynchronous notification
TV Channel Selection User Interface

- Display area
- Search and preview
- Interest specification
View Customization

- **Problem:**
  - How to render multiple channels for simultaneous viewing (aka. display management)

- **Solution:**
  - Automatic screen layout algorithm that considers
    - Visual effects (visually pleasing)
    - Screen utilization
Efficient Media Streaming

- **Problem:**
  - How to ensure
    - efficient network resource usage when there are large number of video streams
    - best user experience given bandwidth constraint

- **Solution:**
  - Semantic-aware bandwidth allocation among channels
  - Locality-aware P2P media streaming
Conclusion

● Opportunities: Sharing Measurement Data/Information
  ● We have posted selected snapshots from the PPLive crawler study on our website http://cairo.cs.uiuc.edu/~longvu2

● Issues:
  ● Content rights management – permission to have TV content on IPTV
  ● Encoding schemes – too many video formats, need to agree on certain TV content format (MPEG2 ?)
  ● Home network content delivery – people are connected via very different networks
  ● QoS/resource management – people will want TV quality on IPTV channels in near future.
Evaluating SIP Proxy Server Performance

Erich M. Nahum, John Tracey, and Charles P. Wright
IBM T.J. Watson Research Center
Hawthorne, NY, 10532
{nahum,traceyj,cpwright}@us.ibm.com

ABSTRACT
SIP is a protocol of growing importance, with uses for VoIP, instant messaging, presence, and more. However, its performance is not well-studied or understood. In this paper we experimentally evaluate SIP proxy server performance using micro-benchmarks meant to capture common SIP proxy server scenarios. We use standard open-source SIP software such as OpenSER and SIPp, running on an IBM BladeCenter with Red Hat Enterprise Linux and Gigabit Ethernet connectivity.

We show performance varies greatly depending on how the protocol is used. Depending on the configuration, throughput can vary from hundreds to thousands of operations per second. For example, we observe that the choice of stateless vs. stateful proxying, using TCP rather than UDP, or including MD5-based authentication can each affect performance by a factor of 2–4.

Categories and Subject Descriptors
C.2.2 [Network Protocols]: Applications; C.5.5 [Computer System Implementation]: Servers; D.4.8 [Performance]: Measurements

General Terms
Measurement, Performance, Experimentation

Keywords
SIP, Servers, Performance, Experimental Evaluation

1. INTRODUCTION
The Session Initiation Protocol (SIP) is an application-layer control protocol for creating, maintaining, and tearing down sessions for various types of media, including voice, video, and text. SIP is of growing importance, as it is being used for many media-oriented applications such as Voice over IP (VoIP), voicemail, instant messaging, presence, IPTV, network gaming, and more. It is also the core protocol for the IP Multimedia Subsystem (IMS), the basis for the 3rd-Generation Partnership Program (3GPP) for both fixed and wireless telephone networks. SIP relies on an infrastructure of servers, which are responsible for maintaining the locations of users and forwarding SIP messages across the application-layer SIP routing infrastructure toward their eventual destinations.

The performance of these SIP servers is thus crucial to the operation of the infrastructure, as they can have a primary impact on the latency of media applications, e.g., for initiating a phone call. However, SIP server performance is not well-studied or understood. Service providers clearly require performance information to understand how to provision their infrastructures to provide reasonable QoS.

The goal of this paper is to shed more light on how SIP proxy servers perform under various configurations and explain some of the limits to performance. We evaluate server throughput and latency for common SIP proxy server configurations, using micro-benchmarks on a dedicated experimental testbed. In particular, we are interested in identifying the primary factors that determine SIP proxy server performance including:

- Where is the time spent in servicing SIP requests?
- How significant are security costs such as authentication and encryption?
- How does the choice of stateless vs. stateful proxying affect performance?
- What is the impact of the transport protocol on performance?

We study these issues experimentally with standard open-source SIP software. We use a common SIP proxy server, Open SIP Express Router (a.k.a. OpenSER), running on an IBM BladeCenter with a 3.06 GHz Intel Xeon. The blade runs Red Hat Enterprise Linux 4 update 3, with a 2.6.17.8 kernel. Performance is measured by clients using the SIPp workload generator sending requests over a private copper Gigabit Ethernet. We measure throughput, average response time, and distributions of response times for a given load level, driving the system not only to capacity but into overload as well.

We find that SIP performance, in terms of throughput, can vary by an order of magnitude, depending on how the server is configured. Proxying throughput can vary from hundreds to thousands of operations per second, depending on whether authentication is used, whether transactions are
2. PROXYING SCENARIO

In this Section, we describe the common SIP proxy server scenario that we evaluate. Proxying is the core SIP function of forwarding a SIP message towards its eventual destination in the SIP infrastructure. In this section, we describe 4 potential scenarios: stateful vs. stateless proxying, both with and without authentication.

Figure 1 shows an example of stateful proxying without authentication. The hashed circle around the proxy illustrates that this is the component ("system under test") that we are measuring. In this example, the first SIP client wishes to establish a session with the second SIP client and sends an INVITE message to the proxy. Since the proxy is stateful, it responds with a 100 TRYING message to inform the client that the message has been received and that it need not worry about hop-by-hop retransmissions. It then looks up the contact address for the SIP URI of the second client and, assuming it is available, forwards the message. The second client, in turn, acknowledges receipt of the message and informs the proxy that it is notifying the user via the 180 RINGING message. The proxy then forwards that message to the initiator of the INVITE, informing the client that the end host has received the message and that the line is "ringing." The user on the second client machine then accepts the call, generating a 200 OK message, which is sent to the proxy which forwards it on to the first client. The first client then generates an acknowledgment. Having established the session, the two endpoints communicate directly, peer-to-peer, using a media protocol such as RTP [10]. However, this media session does not traverse the proxy, by design. When the conversation is finished, the first user "hangs up" and generates a BYE message that the proxy forwards to the second user. The second user then responds with a 200 OK which is forwarded back to the first user.

The above example is for a transaction-stateful, dialog stateful scenario where all SIP messages are routed through the proxy, using the Record-Route: header option. SIP proxies may be configured so that not all messages need traverse the proxy. For example, the BYE/OK exchange could be sent directly between the two clients. The above scenario is frequently used, however, since it enables per-call accounting and billing.

The other 3 proxying scenarios are straightforward extensions of the first.

In the stateless proxying scenario without authentication, the call flows are the same as in Figure 1, except that there is no 100 TRYING message sent from the proxy to the client. In this case, the proxy does not create local state based on the transaction and relies on the endpoints to retransmit lost messages.

In the stateful proxying scenario with authentication, after receiving the initial INVITE, the proxy responds with a 407 UNAUTHORIZED message, challenging the client to provide credentials that verify its claimed identity with a response based on that challenge. The client then retransmits the INVITE message with the generated credentials in the Authorization: header. The proxy also challenges BYE requests, requiring the UAC to retransmit the BYE with the proper Authorization: header, to prevent unauthorized hang-ups.

The stateless proxying scenario with authentication is similar to the previous scenario, except that there is no 100 TRYING provided to the client and no transaction or dialog state created on the proxy.

Because SIP allows the use of multiple transport protocols, including UDP, TCP, SCTP, and SSL, we also wish to evaluate the impact of the choice of transport on performance. In our experiments, we evaluate UDP and TCP. In the case of UDP, all requests and responses are routed through a single connectionless UDP socket. With TCP, each client machine uses a separate persistent TCP connection to the registrar, proxy, or redirector, as appropriate to the scenario.

3. EXPERIMENTAL TESTBED

In this Section we describe the software and hardware utilized in our experiments.
3.1 SIP Server Software

We use the Open SIP Express Router version 1.1.0 (OpenSER) [12], a freely-available, open source SIP proxy server. OpenSER is a “fork” of SIP Express Router (SER) [3], sharing much of its code base. Both proxies are written in C, use standard process-based concurrency with shared memory segments for sharing state, and are considered to be highly efficient. Janak’s thesis [4] describes many performance optimizations that are utilized in SER (and by implication, OpenSER). Each proxy has large feature sets, considerable user bases, active mailing lists, and third-party contributions (e.g., from sip.edu and onsip.org). We chose OpenSER over SER due its better documentation, but we believe our results will hold with SER as well.

In configurations where a user database was required, we use MySQL [11] 4.1.12-3.RHEL4.1, which we populated with 10,000 unique user names and passwords, OpenSER is configured to use a write-back caching policy, to maintain client state across restarts but also to achieve close to in-memory DB performance.

3.2 SIP Client Workload Generator

We use the SIPp [2] SIP workload generator, another freely available open-source tool. SIPp allows a wide range of SIP scenarios to be tested, such as user-agent clients (UAC), user-agent servers (UAS) and third-party call control (3PCC). SIPp is also extensible by writing third-party XML scripts that define new call flows; we wrote new flows that were not included with SIPp to handle authentication. SIPp has run-time options we took advantage of, such as multiple transport (UDP/TCP/TLS) support; MD5-based hash digest authentication, and scriptable support to allow calls to be generated from a list of users.

We made several modifications to SIPp to improve its performance, so as to reduce the amount of client resources required to drive the server to saturation. Most importantly, we removed a restriction that limits the number of outstanding calls to three times the requested load level. This artificially limits the offered load, effectively making SIPp a closed-loop workload generator. Since we are interested not only in the maximum capacity of the server but also how well it behaves under overload, we removed this limitation in SIPp. Our improvements have been contributed back to the open source community and have been incorporated into the latest development releases of SIPp.

3.3 Client and Server OS Software

The servers in our experiments use RedHat Enterprise Linux AS Release 4 update 3, using a locally-built Linux kernel 2.6.17.8, which is more recent than the 2.6.9 kernel variant that ships with RedHat. Our client machines use the SuSE SLES 9 release 2 Enterprise distribution, with a 2.6.5-9 kernel.

3.4 Hardware and Connectivity

The server hardware used is an IBM blade server residing in an IBM BladeCenter. The blade has 2 Intel Xeon 3.06 GHz processors with 4 GB RAM and 100 GB Toshiba MK4019GAXB ATA disk drives. However, for our experiments, we only use one processor. The blade has 2 Broadcom NetXtreme BCM5704S Copper Gigabit interfaces; each interface is connected to a separate Nortel Gigabit switch that is included with the BladeCenter. One switch is connected to our building’s regular LAN, while the other is connected to our private experimental network. To minimize experimental perturbation and variability, all of our measurements are conducted over the experimental network, where minimal other traffic occurs (e.g., spanning-tree). Also residing on the private experimental network are 10 client machines used for load generation; half as UACs and half as UASs. Each client machine is an IBM Intellistation with a 1.7 GHz Intel Pentium 4 processor, 512 MB of RAM, an 18 GB SCSI disk, and an Intel E1000 Gigabit Ethernet adapter.

3.5 Experiments and Metrics

In our experiments, we wish to measure both throughput and latency as a function of load on the server. Throughput is relatively straightforward to define, in terms of the number of the appropriate completed operations per second.

Latency is defined as the time between when the INVITE is sent and the eventual successful 200 OK is received. This is the latency as perceived by the user for initiating a call, which we believe is of more interest than latency that includes the call duration or termination (i.e., BYE). This is similar to the Session Request Delay (SRD) as defined in [5], except that there the latency timer is stopped when a 180 RINGING response is received. Since there is no programmed delay between the 180 RINGING and the BYE on the UAS, we believe the difference is minimal.

For each metric (throughput, latency, and CPU profile) that we report, the number is the average over 5 runs. Latency and throughput curves include 95th percentile confidence intervals. Each run lasts for 120 seconds after a 5 second warm-up time. We also show the cumulative distribution function (CDF) of response times for various load levels, to illustrate how response time varies with load, particularly at 95th and higher percentiles.

3.6 Restrictions, Limitations, and Scope

Note that our setup by no means covers the entire space of configurations for SIP. Results for two major SIP server scenarios, registration and redirection, are excluded due to space limitations, but are available in a companion technical report [6]. We do not consider non-VoIP scenarios such as Instant Messaging or Presence. In addition, there are many VoIP situations not measured by our experiments, including outbound proxying, PSTN gatewaying, ENUM processing, SSL and SCTP as a transport layer, or error processing for unregistered or unauthenticated users. Each of these presents opportunities for future work.

4. RESULTS

Before detailing our proxying results, we believe it is necessary to mention a significant performance fix for transaction-stateful proxying that influences many of our results.

While examining the CPU profiling results generated using oprofile, one problem we observed very quickly was the extremely large amounts of CPU cycles spent in the OpenSER module responsible for transaction-stateful processing. Figure 2 shows an example of this, in the left-hand bar marked “Original.”

Looking more closely, we saw that this time was coming from a single function, insert_timer_unsafe(), which inserted new transactions into a timer structure for retransmissions in the future. This list is sorted by expiration time, yet the routine needlessly searched through the list even
Though the timer needed only to be appended to the end of the list. This function becomes a bottleneck because, at high loads, each new call results in two new transactions (the `INVITE` and the `BYE`), each of which requires a timer to be set and canceled in the common case. This bug is also present in the released version (0.9.6) of SER.

A one-line fix corrected this problem, increasing the peak throughput by over a factor of ten in the transaction-stateful scenario without authentication, from 400 to 4012 calls per second. Performance increases by 250 percent for the scenario with authentication, from 300 to 701. Figure 2 also shows the CPU profile for the same load after applying our fix, in the right-hand bar marked “New.” All results reported below include our fix.

**Throughput**

Figure 3 shows throughputs versus offered load for stateful and stateless proxying, with and without authentication, using both UDP and TCP as transport protocols. Note that both X and Y axes are in log scale. Peak throughputs for each curve are also reported in Figure 5. Peak throughputs are calculated as the maximum throughput achieved while maintaining at least a 99 percent success rate.

As can be seen in Figures 3 and 5, the achieved throughputs vary considerably, depending on how the systems are configured. Starting with the results for stateless proxying with UDP and no authentication as a “best case,” we can illustrate how the various features and functions influence performance.

The most significant feature that influences performance is whether authentication is used. Depending on the configuration, enabling authentication can reduce performance anywhere from 60 percent (in the stateful TCP case) to 90 percent (in the stateless UDP case). CPU profiles for these tests, given in Figure 4, illustrate why performance degrades with authentication. Observe that when authentication is enabled, the profiles show almost half the cycles are spent in the MySQL database and the standard C library functions. Neither of these components are significant when authentication is not enabled; thus, we attribute the C library usage to MySQL. The actual MD5 hash calculation, shown in the profile under the ‘Crypto’ heading, is typically less than 1 percent. The reason is that the database
Figure 6: Average Response Time: Proxying

Figure 7: Response Time CDF: Stateful Proxying, UDP, with Authentication

is consulted much more aggressively when authentication is used, even though OpenSER is configured to use a write-back caching policy, as described in Section 3.1. A straightforward solution would, of course, be to locate the DB on a separate machine, but that would be increasing the resources available, and we wish to study the performance limits of a single node in this work.

The next most significant performance feature is which transport protocol is used, TCP or UDP. Using TCP can reduce performance anywhere from 43 percent (the stateful proxying scenario with authentication) to 65 percent (stateless proxying without authentication). Looking at Figure 4, one can see that the time spent in the OpenSER core goes up significantly, and that the time spent in the kernel almost doubles. TCP is a much more complex protocol than UDP, providing much more functionality, and thus requires significantly larger code paths.

Finally, we see that the choice of stateless vs. stateful processing can also have a significant impact on performance, depending on the configuration. Enabling stateful processing can reduce performance by as much as 60 percent (for the proxying configuration using TCP with no authentication) to having effectively no impact on performance (in the configuration using TCP with authentication).

Observe also that OpenSER does not preserve throughput under overload, as achieved throughput falls quickly when load exceeds the capacity for that configuration. Ideally, a system should maintain maximum throughput even when subjected to overload; this is difficult to achieve in practice, of course, and is the subject of active research. This demonstrates that overload management and control are issues in OpenSER for the future.

Latencies

Figure 6 shows average response times versus load. Note that both X and Y axes use log scales. SIPp has a 1 millisecond timer granularity; thus, any responses that occur within less than a millisecond are treated as zero. Thus, many latencies are not observable on the graph until the load on the server approaches its maximum capacity. At those points, latencies rise rapidly, but the slope of the response times changes once the server is in an overloaded state. Recall that response times are only tracked for successful calls.

Figures 7 and 8 shows the cumulative distributions of the response times measured at several loads for two sample configurations: stateful proxying using authentication with UDP and TCP, respectively. Note that the X axes are in log scale. An obvious and expected result is that, as the loads increase, the response times increase as well (i.e., the curves shift to the right on the graph). There are, however two other interesting features of the graphs.

First, curves tend to cluster in two clearly different regions of the Figures: One, towards the upper left of the graphs, and other, closer to the center and lower right. The characteristic that differentiates these two regions is whether the loads are below or above capacity, i.e., whether the system is under overload. We can see that, when overloaded, the response time distributions become significantly worse, and very quickly (i.e., not linearly in proportion to the load). For example, the stateful UDP auth configuration has a peak throughput of 700 calls/second, yet the gap between the 700 curve and the 800 curve is significant, especially considering the log scale. The TCP curve exhibits a similar gap between 400 and 500 calls/second, as the TCP configuration peaks at 400 CPS. This response time behavior is particularly important for SIP servers, which need to provide service quickly and smoothly, as they are used for real-time media such as voice and video.

Second, observe that several significant jumps occur in the UDP curve at certain response times (e.g., 64 ms, 500 ms, 1000 ms, etc.). The TCP curve, however, does not exhibit this behavior and is much smoother. These jumps are due to the various retransmission timers used by SIP for reliability when UDP is used as the transport protocol. SIP’s primary packet retransmission timer, called Timer A, uses an exponential backoff starting at 500 milliseconds and doubles each subsequent time that it fires. When the system is overloaded, we see the manifestations of these timers firing by the jumps in response time at those timer values. This is because when SIP runs over TCP, it uses TCP’s packet reliability and retransmission mechanisms rather than its own Timer A as is done with UDP. However, higher-level timers, e.g., the transaction timeout timer, are used with
both transport protocols.

5. RELATED WORK

Due to space limitations, we only briefly discuss related work in the SIP server performance area.

Janak’s Thesis [4] describes many of the performance optimizations that are used by SER (and by implication, OpenSER). For example, rather than using zero terminated strings as defined by the C language, SER uses counted strings where the length of each string is stored with the string, making many operations constant time rather than linear based on the length of the string. SER also takes advantage of UTF-8 encoding to canonicalize certain headers for comparing in linear time, despite SIP’s requirement to be case-insensitive. Finally, SER uses lazy parsing to only parse those headers necessary rather than naively parsing all headers, and incremental parsing to only scan needed fields within a header.

Salsano et al. [9] present an experimental performance analysis of SIP security mechanisms using an open source SIP proxy implemented in Java. They found that adding digest authentication to an INVITE transaction increases processing overhead by about 80% for a stateless proxy and 45% for a stateful proxy. They found minimal overhead using TCP or TLS instead of UDP. The server performance in these experiments was on the order of tens of calls per second. Given these relatively low numbers, we are not certain how representative these results are.

Cortes et al. [1] measured the performance of four transaction stateful SIP proxies using a suite of five tests. Proxy performance ranged from 90 to 700 calls per second. The tests used UDP only and evaluated parsing, string processing, memory allocation, thread overhead and overall capacity. Their results showed each of these components significantly affected performance, with parsing, string handling and memory management contributing from 33% to 88% of processing time.

6. SUMMARY

In this paper, we evaluate SIP proxy server performance, examining the impact of authentication and transport protocol on performance, as well as statelessness vs. statefulness. We study these issues experimentally, using OpenSER, a high-performance open-source SIP server, and SIPp, the de-facto standard for SIP performance benchmarking.

We find that performance varies widely, by an order of magnitude. Depending on the configuration (authentication enabled/disabled, UDP or TCP, stateful or stateless), throughput can vary from hundreds of operations a second to thousands. Authentication has the greatest impact across all configurations, due to the increased use of the database. TCP is more expensive than UDP for most configurations, and stateful proxying slower than stateful proxying. We show that latency distributions are highly influenced by the load, especially when the system is in an overloaded state.

Based on our results, we believe many potential future research issues exist, including examining other SIP server scenarios, optimizations identified by profiling, overload control, and using SSL/TLS as a transport.

The full scope of this work is available as an IBM Research Report [6]. A short 2-page extended abstract appears in [7].

Acknowledgments

Thanks to Arup Acharya and Henning Schulzrinne for many insightful discussions, and to Vijay A. Balasubramiyan for help with configuring OpenSER.

7. REFERENCES

CHARACTERISING USER INTERACTIVITY FOR SPORTS Video-on-Demand

Andrew Brampton †
Nicholas J.P. Race †
Andrew MacQuire †
Laurent Mathy †
Idris A. Rai †
Michael Fry ‡

†Lancaster University
{brampton,macquire,rai,race,laurent}@comp.lancs.ac.uk
‡University of Sydney
michael.fry@usyd.edu.au

ABSTRACT
This paper presents a detailed characterisation of user behaviour for a series of interactive sport videos from the 2006 FIFA World Cup. In addition to generic VCR-like features, our custom-built Video-on-Demand architecture enabled us to provide advanced interactivity features such as bookmarking. We illustrate how such functionality may have a dramatic impact on how users consume content. A detailed discussion is also provided on how content distributors may turn this knowledge to their advantage, and thus increase the efficiency of their delivery networks.

1. INTRODUCTION
In recent years the Internet has increasingly been used to distribute bandwidth-intensive, low-latency streaming media. Due to the resources required to deliver such content, dedicated Content Distribution Networks (CDNs) are often used to improve the end-user’s experience. As such systems evolve, users expect correspondingly improved interactive functionality; something which is increasingly difficult to achieve with diverse content types exhibiting varied access patterns. In order to provide a high quality of service, modern CDNs must therefore have an in-depth understanding of user behaviour regarding different content types.

A number of previous papers have already studied the characterisation of user behaviour for Video-on-Demand (VoD) content. Some have studied single genres such as educational videos [3, 6, 1], whereas others have examined a range of video types [7, 10, 5, 13]. This paper is closely related to previous work that examined user characterisation for interactive video. Similar studies exist where logs were analysed from VoD systems that support VCR interactivity, i.e. the ability to pause, resume and skip back and forth within a given video stream [7, 11, 12]. The typical approach in these studies has been to analyse either publicly available traces of static content (such as those at the Internet Traffic Archive [8]), or privately obtained logs of more dynamic streaming content from larger networks such as Akamai’s [2].

In this paper, we study user behaviour for an interactive VoD system that serves users specifically with video from the 2006 FIFA World Cup. A key distinguishing element of our work is the fact that we implemented our own VoD system, designed to offer novel interactive functionality beyond typical VCR-like features. The prime example of this is bookmarking: direct links to points of interest within the video. Our system also allowed users to contribute their own bookmarks at any time, distinct from those added during the publishing process. An example of a bookmark within our content could be a common event such as the match kick-off, or a potentially more popular event, such as a goal.

Previous studies making use of entertainment content may have witnessed the classic start-to-finish playback model in their access patterns, with occasional user VCR interactivity. In our experiment, however, sports content proved highly dynamic. Users often chose to watch (and re-watch) small segments of the full video, in a complete departure from the start-to-finish model. The behaviour observed may also be present in other sports, and different content genres (e.g. educational, entertainment, news, etc.), as these genres often have a few popular highlights.

We found that the bookmark functionality in our system had a significant impact on user behaviour, leading to access patterns quite dissimilar from previous related work. We identified distributions, namely log-normal, Weibull and normal, to best model various metrics and workload properties. These models can be used to drive simulations of the type of interactivity behaviour studied in this paper. We also discuss how delivery networks can exploit the observed behaviour to improve user-perceived performance. For instance, we show that the order in which users view bookmarks can be predicted based on previous activity, enabling CDNs to leverage this data for performance gains.

The remainder of this paper is structured as follows: Section 2 describes our experimental setup, while Section 3 analyses the results. Section 4 concludes the paper with a discussion of the impact of our results on existing Content Distribution Networks.

2. EXPERIMENTAL SETUP
We set up a simple, interactive video-on-demand system. The system was divided into three main components: the capture server, the Video-on-Demand server, and a web interface.

1More information about the system and its source code is available at http://www.rcdn.org/
Figure 1: Video-On-Demand Interface

Our capture server recorded public broadcasts of raw MPEG-2 streams from the beginning of the pre-match commentary through to the end of coverage. Once a full match was captured, the system transcoded the stream to high and low bitrate Macromedia Flash 7 FLV streams (1 Mbps and 300 Kbps respectively). Administrators would then manually add metadata to the system describing the location of key events within the match. These locations are referred to as bookmarks, and typically included events such as the beginning and end of the match, any goals, and other important events such as red cards. The final FLV streams were then transferred to the VoD server. This full process typically took 6 hours and thus the matches were available shortly after being played.

The VoD system was an Apache webserver, which served the Flash based user interface over HTTP. This server was only accessible to staff and students within Lancaster University’s campus, and those connecting remotely via VPN. To aid in logging, the user interface would make its HTTP requests as verbose as possible, allowing us to later track users through multiple sessions and determine exactly which controls were pressed and when. Additionally, each playback window would maintain a periodic HTTP-request heartbeat with the server, which was used to determine when connectivity was unexpectedly lost.

The web interface consisted of two main sections; firstly a index page allowing the user to select any available match from the World Cup, and secondly the player interface that displayed the video of the matches, as shown in Fig. 1. The player interface offered some simple controls to seek around the match. We were aware that the user interface would constrain the user’s actions, and it was therefore designed to be as simple and generic as possible. Forward and backward buttons were provided that allowed seeking 10, 30 and 60 seconds in either direction. These only accounted for rather small jumps, and so we also provided a seek bar which enabled users to make potentially large jumps to any arbitrarily chosen time. Finally, users had the list of administrator-added bookmarks which enabled them to seek directly to key events. This interface was also extended to allow users to submit their own bookmarks (via the tag button), which other users could see and select. These user bookmarks covered events that were not typically bookmarked, but were of particular interest (such as events that came under later scrutiny).

<table>
<thead>
<tr>
<th>Action</th>
<th>Occurrences</th>
<th>Percentage($)</th>
<th>per session $</th>
</tr>
</thead>
<tbody>
<tr>
<td>Back 10s</td>
<td>1643</td>
<td>7.24</td>
<td>0.79</td>
</tr>
<tr>
<td>Back 30s</td>
<td>556</td>
<td>2.46</td>
<td>0.24</td>
</tr>
<tr>
<td>Back 60s</td>
<td>775</td>
<td>3.43</td>
<td>0.33</td>
</tr>
<tr>
<td>Forward 10s</td>
<td>3319</td>
<td>14.67</td>
<td>1.42</td>
</tr>
<tr>
<td>Forward 30s</td>
<td>1684</td>
<td>7.36</td>
<td>0.71</td>
</tr>
<tr>
<td>Forward 60s</td>
<td>3488</td>
<td>15.42</td>
<td>1.49</td>
</tr>
<tr>
<td>Seek-bar</td>
<td>2101</td>
<td>9.29</td>
<td>0.90</td>
</tr>
<tr>
<td>Bookmarks</td>
<td>5293</td>
<td>23.00</td>
<td>2.22</td>
</tr>
<tr>
<td>User bookmarks</td>
<td>585</td>
<td>2.59</td>
<td>0.25</td>
</tr>
<tr>
<td>Add bookmark</td>
<td>43</td>
<td>0.19</td>
<td>0.02</td>
</tr>
<tr>
<td>Pause</td>
<td>1847</td>
<td>8.16</td>
<td>0.79</td>
</tr>
<tr>
<td>Resume</td>
<td>1690</td>
<td>7.47</td>
<td>0.72</td>
</tr>
</tbody>
</table>

Table 1: Number of occurrences of interactive actions

3. ANALYSIS

The 2006 FIFA World Cup ran from the 9th of June until the 9th of July, whereas the results analysed in this paper were recorded from the 13th of June until the 16th of July. The data for the first 4 days was discarded due to alterations made to the logging system and user interface in that period. Also, the 7 extra days considered after the end of the World Cup were added due to continued use of the site. A total of 66 matches were logged (64 from the event, and 2 pre-competition friendlies), with 405 unique users over the one month period. On average 30.7 unique users viewed each match.

In this section, we use the logs from our experiment to characterise user properties for our system. R-Square fitting is used to determine models for the various features analysed, such as popularity, session length, inter-seek times, and bookmark requests.

3.1 Interactions

Recall that our system allowed various interactive operations, namely pausing, resuming, seeking forwards & backwards, and jumping to bookmarks. This range of operations, combined with the nature of the content, highly influenced user behaviour. For most users, there was a complete departure from the typical start-to-finish playback model that has often been observed in previous work [7].

Table 1 shows the number of times each action was taken and its corresponding percentage against all other operations. Forward seeking was used a combined 37.44% of the time, whereas backward seeking was only used 11.86%. These actions only accounted for the relatively small jumps (10, 30, and 60 seconds), whereas potentially large jumps (seek-bar and following bookmarks) made up 34.87% of all operations. The table also shows that, on each session (viewing of a single video), a user on average used backward actions 1.15 times, bookmarks and seek bar actions 3.37 times, and 3.62 times for forward actions.

Previous studies have shown that the most common action is pause/resume [7], however we see that for our traces, forward operations are by far the most common, closely followed by seeking to bookmarks. The table also shows that the number of pause operations account for 8.16% of all actions. Our reduced number of pauses can be explained by the short session durations observed. This is in accordance with previous work which found a positive correlation between session time and the number of pause operations [12].

To get a detailed picture of how users navigate through a bookmarked video, we analyse the behaviour of users within a single match (Argentina vs. Serbia and Montenegro). This match had 10 bookmarks, 3 user defined bookmarks, and was the most popular match of our experiment in terms of the number of viewers, although the user behaviour wit-
<table>
<thead>
<tr>
<th>To time (seconds)</th>
<th>From time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1000</td>
<td>0</td>
</tr>
<tr>
<td>2000</td>
<td>1000</td>
</tr>
<tr>
<td>3000</td>
<td>2000</td>
</tr>
<tr>
<td>4000</td>
<td>3000</td>
</tr>
<tr>
<td>5000</td>
<td>4000</td>
</tr>
<tr>
<td>6000</td>
<td>5000</td>
</tr>
<tr>
<td>7000</td>
<td>6000</td>
</tr>
<tr>
<td>8000</td>
<td>7000</td>
</tr>
</tbody>
</table>

**Figure 2: Jumps made by users within the Argentina vs. Serbia and Montenegro match**

Vertical and horizontal lines in Fig. 2 represent the 10 bookmarks of the match. Every point is a jump that is identified by corresponding times on both axes. The diagonal line is a present-time marker such that the forward jumps are points which lie above it, while backward jumps appear below it. Thus, no point falls precisely on the diagonal, yet can be close to it. We can immediately observe from the figure that many points fall on horizontal lines, implying that most jumps involved seeking to bookmarks.

The forward jump buttons appear to have been mostly used for browsing, as can be seen between 0 and 1000 seconds. This could be due to user unfamiliarity with the interface, or possibly users first checking for anything interesting at the start of the video before moving to a bookmark. The rewind action is used mostly around events, for example in the case where users wish to re-watch the event, or when the bookmark insufficiently marked the beginning of the event.

A good example of this was before the bookmark at time 2815, where users rewound up to 75 seconds to see more of the build up to the goal. Clusters of points can also be seen on horizontal lines shortly after a vertical line, indicating that users jumped from one bookmark to another. We can also observe large clusters for each vertical line around only one horizontal line, such as from the bookmark at 1300 seconds going to the bookmark at 2815 seconds. This shows that a majority of users visited consecutive bookmarks. In this case users went from the first goal (at time 1300) to the second goal (at time 2815). The results on this single match demonstrate that when presented with bookmarks in sports video, users are highly influenced by them.

### 3.2 Popularity

We study popularity in terms of the number of viewers who watched an object or a segment. An object in our system is a single football match whereas a segment is a section of video one second in length.

The ranking for both object and segment popularity is shown in Fig. 3. Recall that there were only 66 matches recorded, which is why the object rank ends much earlier than for segments. Our analysis reveals that object popularity does not follow the typical power-law distribution observed within CDNs [6, 3, 13] but instead is a normal distribution with parameters $\mu = 33.2$ and $\sigma = 17.1$. This can be attributed to the nature of the World Cup and the relatively few new objects each day.

We found that the popularity of one-second segments in videos exhibit a log-normal distribution with parameters $\mu = 0.0159$ and $\sigma = 1.35$. Note that log-normal distributions closely relate to power-law or heavy-tailed distributions [9]. They are skewed distributions where a small percentage of samples contributes to a sizeable weight of their distribution. We observed that a small percentage, (the 10% most popular segments), accounted for about 44% of all requests. Previously, Costa et al. [7] found that for educational and entertainment content, the popularity of segments is roughly uniformly distributed with a slight skew towards the beginning for entertainment content. Our result, however, implies that there are segments with orders of magnitude more viewers than others. To illustrate this fact, we present Fig. 4, which shows the popularity of each second of video for the Argentina vs. Serbia and Montenegro match. It is very clear from the figure that the bookmarks influence the popularity of segments. We also observe that most of the bookmarks that users found interesting are equally popular, receiving requests from around 60% of all viewers of the match.

Popularity metrics are important to CDNs as they help to decide what resources to allocate to each object. We have seen that bookmarked sports videos provide a content format with specific segments of interest (goals, for example). This result emphasises the use of partial caching techniques to cache only popular segments of objects, as has been previously proposed [4].

**Figure 3: Object and segment popularity**

**Figure 4: Viewers of the Argentina vs. Serbia and Montenegro match**
3.3 Longevity

The popularity of both videos and bookmarks in our system faded over time. We call the duration at which any such item remains utilised its *longevity*. The study of a video or bookmark’s longevity can be used to compute corresponding distributions, aiding in content management decisions for similar items in the future.

Fig. 5 shows the popularity of bookmarks versus the time they were first utilised. The figure suggests that following an initial peak and a slight resurgence, there was a rapid decrease in interest after just a short period.

R-Square fitting reveals that the bookmark longevity can be suitably estimated using a Weibull distribution with \( \lambda = 1.814 \) and \( k = 0.6383 \). This suggests that the popularity exhibits heavy-tailed properties. We also observed that half the bookmark usage occurs within 24 hours, with the remainder slowly occurring over the following 2 weeks.

3.4 Session lengths

Session length is the total time a user spent on a video. A session may consist, for example, of a user watching part of a video, pausing for a while, then continuing to watch the video. Therefore, it is possible that a session is longer than the actual length of a video.

Fig. 6 shows the CDF of both session and inter-seek times (discuss on inter-seek times follows in the next section). It can be observed from the session times that most users access a video for a very short time relative to the length of the overall content (i.e., possibly just watching interesting events from the match). In particular, note that around 80% of sessions lasted less than 500 seconds. Given that a length of a typical video was 2.5 hours long, 500 seconds is only 5.5% of the video. The average session duration is found to be only 10.2 minutes.

Fig. 6 also shows that only a small minority of users (roughly 3%) could have possibly watched the entire match. We found that a small number of users (1.17%) accessed the video for between 3 to 8 hours. Our logs show that these users paused a video for a long time before they decided to resume playback. These outliers are possibly why we observe that session sizes are best fitted by a log-normal distribution with parameters \( \mu = 4.835 \) and \( \sigma = 1.704 \).

3.5 Inter-seek times

Inter-seek time is described as the time a user spent actually watching a section of a video before seeking to a new location (disregarding any paused periods within the session).

From our logs, we found that on average a user performed 9.3 seek operations around a video resulting to a mean inter-seek time of 66 seconds. Fig. 6 shows the CDF for inter-seek times as well as session length. It can be seen that the majority of users viewed the content as a series of excerpts, usually under a minute in length.

We found that inter-seek times can be estimated by a log-normal distribution with parameters \( \mu = 1.4796 \) and \( \sigma = 2.2893 \). Previous studies have also found that the majority of inter-seek times are very short [12]. They have also been shown to be approximately *Poisson or Pareto* for educational content in different servers [3]. A distribution of inter-seek times can be used by a delivery system to determine the size of video replicas and the time that it should react before a user seeks elsewhere on the video.

3.6 Sequence

In this section, we analyse the data we collected from our experiment to study the extent to which user actions on a video can be predicted. We call the order that events in a single match are viewed a *sequence* of events. A typical sequence can include any combination of user actions on a video such as starting the video, VCR actions, seeking to a bookmark, *etc*. Additionally, we separately consider sequences that consist of jumping between bookmarks. If a
system could detect or predict patterns within a sequence, then it could pro-actively respond to them in order to optimise content delivery. For example, a server could use spare bandwidth to push out the appropriate content to a user before being requested, or a client could be allowed to pre-cache content based on popularity of a pattern in a sequence.

We first identified sequences of actions from our traces. We then broke the sequences into pairs of events that were accessed consecutively. For example, a sequence made up of starting the video, jumping from bookmark A to bookmark B, forwarding 30 seconds (F30), and finally ending a session was broken into 4 pairs, namely, Start → A, A → B, B → F30, F30 → End. Note that the event End means the end of a session and not necessarily the end of a video. The number of occurrences of each pair was totalled for each match and normalised by the number of users who watched that match. The ratio of users to each sequence pair is shown as a CDF (for both bookmarks alone and inclusive of VCR actions) in Fig. 7. Note that the x-axis of the figure goes above 1, which shows that some sequence pairs have been followed more than once by some users. Intuitively, the more the sequence pair is followed, the more predictable it is, and thus larger values on the x-axis represent better predictability for a given sequence.

Fig. 7 shows that around the top 20% of bookmark sequence pairs were followed by more than 50% of users. This means that there is a high chance of predicting these bookmark sequences. Note that these bookmarks also consist of the 20% most popular bookmarks. However, the figure also shows that it is generally difficult to predict the actions of a user if all actions are considered. This is because of the wider range of interactivity options a user has when VCR functionality is also considered.

We now summarise the analytical models in Table 2. We found that user metrics could be estimated by more than one distribution, however the table shows only the best fit for each. The table also shows corresponding $R^2$ values that illustrate the suitability of the models. Of particular importance is the type of distribution which can have a significant impact on the system. For example, the Weibull and log-normal distributions are both heavy-tailed, for which systems may have to anticipate the uneven distribution to cope effectively.

4. CONCLUSIONS AND FUTURE WORK

We have presented a study and characterisation of user behaviour for interactive sports video-on-demand (VoD) systems. Using our custom-built VoD system, we captured FIFA 2006 World Cup matches and made them locally available through a highly interactive user interface.

Our results show that the interactivity options available to users highly influence their behaviour. In particular, we found that the novel interactive feature called bookmarking, played a pivotal role, leading to access patterns quite dissimilar from previous related studies that looked at VCR-like interactivity alone. The combination of our content type and the addition of bookmarks led to users accessing content in relatively small segments that were both highly popular and sparsely distributed throughout the length of the videos. These popular segments, more commonly named hotspots, were clearly skewed around bookmarks. From both a user and a CDN’s perspective, this can be viewed as advantageous; users can reach interesting content more quickly through the bookmarks, and the increased locality of interest means the CDN can respond more effectively.

Interest in a given event is subjective, however, and an administrator or individual user’s opinion inevitably plays a major part in the optimality of manually placed bookmarks in terms of their locations and validity. For delivery networks to benefit from the user characterisation observed in this paper, it is clear that they must be optimised to allow for autonomic repositioning of incorrectly placed bookmarks, detection of hotspots, and caching/replication, all based on predicted or past recorded user behaviour.

A particular section of a piece of content may prove to be far more popular than the remainder, thus making it suitable for special treatment. Service providers can either rely on their users or administrators to add appropriate metadata describing potential hotspots on the video. However, neither would necessarily know how demand for their content would change over time. As such, reactive and adaptive approaches may prove most suitable for the general case, where the evolution of hotspots and the validity of bookmarks are autonomically detected based on observed patterns.

A simple way of achieving this would be to select a threshold (e.g. a percentage of users) and classify any section of video which exceeds it as a hotspot. This method may take time to identify the correct hotspots since requests must first be recorded across the length of a video. The system could then mark an identified hotspot with a bookmark for easy access to future users. Additionally, longevity information can be used in conjunction with detection algorithms to decide upon the validity of existing bookmarks.

Upon examination of our logs, we found that some of the bookmarks were incorrectly placed. Users who discovered that the bookmark started sooner or later than they expected are likely to make a corresponding jump shortly after requesting a bookmark (as observed in Fig. 2). We observed that 40% of the bookmarks had at least one user who
jumped to a position before the bookmark itself. Upon inspection of the video content, however, we discovered that when users consistently made an additional seek, it was typically because bookmarks were placed during the run-up to a penalty kick, omitting the cause of it. Users wanting to see the relevant incident would therefore have to seek backwards, as the bookmarks were incorrectly placed beyond the beginning of the actual hotspot.

We also analysed the distance from a bookmark that users jumped shortly after requesting it (within a range of 200 seconds either side, as to ensure relevancy to the bookmark). An examination of the traces revealed that approximately 84% of the seeks considered were carried out within 20 seconds of moving to the bookmarks, perhaps representing the users who were almost immediately dissatisfied with the bookmark location. Fig 8 shows the varying distances from the bookmark that users moved to, all within 20 seconds of reaching the bookmark's location. We observe from the figure that around 6% of bookmark requests have seeks shortly afterwards, possibly due to suboptimal bookmark positioning. It was noted that 5.7% of all bookmark requests were for penalties, adding support to this theory.

Placing bookmarks optimally has the advantage of improving user experienced performance and reducing load on servers, whereas incorrectly positioning bookmarks leads to an increase in the number of seeking requests. It is therefore desirable for a delivery system to be capable of repositioning relevant bookmarks accordingly to enable faster access of relevant content, and the resultant reduction in seek operations would reduce network load.

We experimented with a repositioning mechanism based on a exponentially-weighted moving average algorithm, using our logs as input. We observed that the algorithm successfully identified the correct positions for poorly positioned bookmarks and left the positions of well positioned bookmarks unchanged (results are omitted due to lack of space). In our future work, we will check to see if such an algorithm can perform similarly in delivery networks and we will devise new algorithms if necessary.

Beyond issues related to their placement, the resultant effect of bookmarking was that several sections of each video were far more popular than others. In a CDN context, this reaffirms the idea that content should not be treated as immutable objects, and instead it should be divided into small segments. A network can then rank the segments in terms of their popularity and apply caching or replication strategies accordingly.

Naturally, the size of the segments in question is an important concern. In this paper we have examined the content on a second-by-second basis; while this may seem a small value given that all our videos were several hours long, Fig. 6 indicates that 60% of all sustained playback operations were actually 10 seconds or less, and thus small segment sizes would potentially be required.

Developing algorithms to predict the actions of newly arriving users based on the experience of past users may also prove useful. For example a CDN could infer the order of segments users typically viewed, or the probability of viewing any given segment after another. This could form a probability matrix with each destination weighted by the percentage of previous users who chose that destination; knowledge which may then be useful to both the server and the client. On the server side, the server would be aware of which content to pre-fetch from the original source or pre-push to other servers or clients in advance. This could also influence caching eviction decisions whereby objects are evicted based on their usage as well as the popularity of objects likely to be requested with them.

Future work may involve expansion upon all of these concepts, along with further characterisation of differing content types.

5. REFERENCES
Scalable Application-Specific Measurement Framework for High Performance Network Video

Congxiao Bao  
Network Center of Tsinghua University, Beijing, 100084, China  
86-10-62603020  
congxiao@cernet.edu.cn

Xing Li  
Network Center of Tsinghua University, Beijing, 100084, China  
86-10-62785983  
xing@cernet.edu.cn

Jinpeng Jiang  
Dept. of Electronic Engineering of Tsinghua University, Beijing, 100084, China  
86-10-62602935  
Jjp02@mails.tsinghua.edu.cn

ABSTRACT
Extensive studies show that bandwidth provisioning is the key method to guarantee the performance of mission-critical real-time network applications. Unfortunately, network congestions, rate-limiting for special addresses and ports, and even the bad cabling may result in the bandwidth bottleneck along the end-to-end path. This is the main cause of degradation of the application performance, especially for the high-performance video applications which consumes large-bandwidth. Our experiences, being both as a national academic Internet Service Provider and end-user of high-performance video applications, clearly indicate that using suitable measurement tools are critical to find the network bottlenecks and enable a successful video session.

To support high-performance video on the Internet, taking the popular Digital Video Transport System (DVTS) as an example, we propose a Scalable Application-Specific Measurement (SASM) framework. The SASM for DVTS involves two novel components. One is a set of DVTS specific network and application measurement tools, which provide a common understanding for both end-users and network administrators. The other is the scalable management scheme for distributed DVTS-specific measurement servers, including server auto-discovery and server auto-registration. Deploying SASM on several academic backbones indicates the effectiveness of the proposed framework for the troubleshooting and network fine-tune of high-performance video applications.

Categories and Subject Descriptors
C.2.3 [Computer-Communication networks]: Network Operations

General Terms
Management, Measurement, Performance, Design, Experimentation

Keywords
High performance network video, application-specific measurement, measurement server auto-registration, measurement server auto-discovery

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

1. INTRODUCTION
More and more demonstrations and trials of real-time high-performance video applications on the Internet have been carried on for recent years. Digital Video Transport System (DVTS) [16] is the most attractive and useful one among those systems. With relative high-resolution (720*480 pixels), reasonable latency and acceptable setup cost, DVTS is widely used for global distance-learning on telemedicine [10], music-teaching and art learning on telemedicine [10], music-teaching and art learning on telemedicine [10], music-teaching and art learning. DVTS system even for a point-to-point connection is a big challenge for both Internet Service Providers (ISP) and end-users, especially for inter autonomous system (inter-AS) collaboration.

Besides the performance problems described in [2], there are several factors which make the 30Mbps transmission difficult on the Internet.

(1) The link bandwidth is limited and the link utilization is high in some segments of the packet traveling path.

(2) The network administrators may setup firewalls and QoS features to suppress the large throughput for specific protocols, IP addresses, port numbers, packet size and even the application headers for security reasons.

(3) Ethernet devices may have bad cabling and/or wrong duplex mode setup.

It is clear that a single breakpoint on the video traveling path will cause degradation of application performance or even destroy the application session. Furthermore, network behavior may change between two events or even two rehearsals. It is possible for the network administrator to configure the BGP routing to guarantee the bandwidth provisioning for one specific case, but it may not be suitable for others. Doing routing adjustment case by case also increases the possibility of wrong configuration by the administrators when a lot of requests arrive in short period.

In addition, setting-up high-performance video applications on Internet like DVTS is a complex process which involves network protocols, application protocols and human protocols working together [1]. The end-to-end video performance is often plagued by application, operating system and network problems interacting together. Heavy load of negotiation among end-users, application system engineers and network administrators, time differences and sleepless working for all-sites are the non-technique factors which make the test painful.

Existing measurement tools for troubleshooting the high performance video applications are mainly divided into two categories, the network performance measure tools and the...
application-related trouble-shooting tools. Usually, the network performance measurement tools, like ping, traceroute [8], pathping [20], AMP/iperf [9] [11] [17] etc. measures network performance using different protocol, port numbers and traffic patterns from the real video applications. Therefore, the measurement statistics based on these tools may not reflect the real network and application evidence. H.323 Beacon [2] [3] is a pioneer of application-related troubleshooting tool. It adds additional features sufficed in the context of H.323 application, however, the bandwidth requirement of H.323 video application is less than 3 Mbps. Our experiences also indicate that the main problems of H.323 video application occurred in the “last mile”, (i.e. the firewall and end-system configuration), while the main problems of DVTS application occurred in inter-AS borders under complicated routing policies. Furthermore, most of the existing tools lack the ability for a scalable management scheme for large-scale distributed measurement. The studies in [12][13] outline the large-scale distributed information management scheme based on a mathematic model, but those descriptions only give general guidelines for the design and implementation of distributed contents without specific technologies on distributed servers management in detail. To meet the high-performance video application requirements, we do need suitable measurement tools. The design goals of such tools include:

(1) The measurement tools should be able to simulate the real video application characters, i.e. the protocol type, the source and destination port number and the traffic patterns, in order to provide the real network and application evidence for troubleshooting and isolate the application problems from network problems.

(2) Since the hop by hop performance guarantee is the basis for end-to-end performance, the measurement server should be placed alone the packet traveling path. So, the end-to-end performance problems can be decomposing into segment or even hop problems. The measurement tools should be able to find these servers and to do the measurement test with each server automatically at any time.

In this paper, taking DVTS as an example, we propose a new measurement framework supporting large-throughput real-time video applications, called the Scalable Application-Specific Measurement framework (referred to as SASM in the rest of the paper). In SASM for DVTS, we developed a set of DVTS-specific network and application measurement tools (dvping, dvmcast and the existing DVTS application), which can provide a common understanding among end-users, application system engineers and network administrators. A simple but straightforward server management scheme, including measurement server auto-registration, intra-autonomous system measurement and measurement server auto-discovery is also proposed in this paper.

The rest of the paper is organized as follows. Section 2 describes the key components and general framework of SASM. Section 3 introduces the case study of SASM in CERNET [14], CERNET2 [15] and TEIN2 [18]. Finally, concluding remarks and future work are given in Section 4.

2. SASM FRAMEWORK

2.1 Application-Specific Measurement Tool

The SASM uses general client/server architecture. The testing between two segments of the network can be achieved by enabling a communication session of a pair of client-server in the corresponding segments. It is important that the communication socket of a client-server pair for measurement tool must have the same socket parameters (protocol, destination address, destination port, source address, source port, application payload pattern) as that of a real DVTS streaming. As shown in Figure 1, the server called dvmcast has two modes (sending and reflecting) and the DVTS application and dvping are the two types of clients. The dvmcast can either read a DVTS data file [5] [6] and send it to the client, or reflects the received real-time DVTS streaming from a client and send back to the same client. The DVTS application system, which receives the video stream by dvmcast or sends a video stream to dvmcast and receives the reflected video stream, can be used by the end-users to check the video/audio quality subjectively. DVTS file sending on demand also provide a possible way for the network administrators to use and understand what the application users are talking about. The dvping, which sends a simulated DVTS stream with the same UDP protocol, destination address, destination port of DVTS session, source address, source port of DVTS session, DVTS payload pattern [5] to dvmcast and receives the reflected packet stream with packet loss, round trip time, sending and receiving throughputs reporting, can be used by both the end-users and the network administrators to check the network performance. Based on this network and application measurement combination, the end-users and the network administrators can exchange the testing results without the help of the counterpart and understand the problems described either in the network engineering language in terms of loss/throughput/latency or the application language in terms of video/audio quality. Since intentionally or unintentionally sending DVTS stream from dvmcast can generates large traffic congestion, our systems are designed that the streaming can only be send to a permitted hosts. For the same reason, dvping has a special feature, which automatically limits the sending throughput in the bad network condition. Therefore, the DoS attack can be avoided and the impact of the proposed measurement testing to network can be minimized.

![Figure 1 DVTS-specific measurement tool](image)

2.2 Scalable Management Scheme for Distributed Servers

To guarantee the end-to-end video application performance, it is necessary to do troubleshooting segment by segment or hop by hop using distributed servers.
2.2.1 Methodology for the scalability
Currently there are about 1,342,177,280 addresses in the global Border Gateway Protocol (BGP) routing table [4]. This is roughly the theoretical maximum number of hosts on the Internet, which may run end-to-end high-performance video applications. Therefore, we do need a method to handle the scalability problem for the measurement test. It is well-known that the scalability problem can be solved by aggregation. Figure 2 shows the different numbers of possible measurement servers (referred to as MS in the rest of paper) required based on the different aggregations methods on the Internet. The tradeoff is the better aggregation (scalability) or the better measurement accuracy along the path. It is usually true that the network performance bottleneck happens at the autonomous boundary or the access networks, therefore, we chose to pickup at least one measurement server in each autonomous system. Based on this method, the number of measurement servers should be reduced to 20,000, the same as the current number of autonomous systems. Any IP address on the Internet can be mapped to the autonomous system number (referred as ASN in the rest of paper) using the global BGP table.

2.2.2 Intra-AS measurement
In order to follow the rule of “at least one MS in each AS is needed”, the assumption “the network performance is reasonably good inside an autonomous system” have to be ensured. We suggest deploying measurement servers on every possible Pop inside the same autonomous system and propose to run dvmcast and dvping inside each autonomous system periodically. The visualized performance reports of full-mesh tests among Pops with packet loss and latency based on beacon method [7] is called dvbeacon in SASM. The network administrators should be able to monitor the dvbeacon and fine-tune the network in their autonomous system to guarantee the above assumption. Figure 3 gives an example of performance monitoring of dvbeacon inside one autonomous system.

2.2.3 Mapping
Mapping is the process which involves two sub-processes, one is any IP address of each hop obtained by traceroute to the ASN mapping and the other is ASN to measurement server IP address mapping in the same autonomous system. The explanation of two mapping procedures is shown in Figure 4.

The first mapping is based on the global BGP routing table, either obtained from a local BGP speaking router or obtained from global BGP routing table lookup services (e.g. routeviews provided by the University of Oregon’s Route Views Project [19]). For example, in order to find the ASN for IP address a.b.c.d, the DNS query of d.c.b.a.asn.routeviews.org is submitted to DNS system (asn.routeviews.org ) and the corresponding ASN and the prefix (100 a.b.c.0/24) can be retrieved in the text field of the DNS answer. As designed, the measurement server is registered in the database beforehand (see Section 2.2.5) with the domain name and IP address mapping pair, as100.dvts.foo.bar and w.x.y.z respectively, so the second mapping can be achieved based on the ordinary DNS lookup.

2.2.4 Find the measurement servers alone the packet traveling path (auto-discovery)
Finding the measurement servers alone the packet path is most important part for doing the measurement test hop by hop. In SASM, it is assumed that, for identifying the performance segment-by-segment, at least one measurement server should be placed in each autonomous system along the path, as shown in Figure 5.

Since the Internet routing may not be symmetric (the inbound and outbound traffics for a user are not in the same path in opposite direction, as dash line and dotted line for user A shown in Figure 5), measurement tools are designed in such a way that the one way (sending or receiving) and the roundtrip (sending-reflecting-
receiving) tests are both available. Assume that at least one MS has been setup in user’s subnet, like MS_A and MS_B in Figure 5. When user A and user B want to setup a DVTS session between them, user A is able to find outgoing path from A to B by taking the ordinary “traceroute” and the incoming path from B to A via the web service issuing remote command “traceroute A” provided by MS_B. The mapping procedures can then be used to find the autonomous system numbers (ASN) and IP addresses of the measurement servers in each AS alone the path, described in Section 2.2.3 (i.e. user A can get the IP addresses of measurement servers MS_4, MS_3, MS_1 and MS_A on its incoming path and measurement servers MS_1, MS_2 MS_4 and MS_B on its outgoing path). Therefore, the network problem can be identified by users and administrators between any two segments along the paths.

Figure 5 Network topology and measurement servers layout

2.2.5 Measurement server auto-registration
The SASM framework provides a way for newly developed measurement server to do auto-registration in a scalable manner. It includes following steps shown in Figure 6.

(1) Measurement Server Registration
The DNS query method is used to do the registration. The new working MS sends a predefined sequence of DNS query to the registration server (which is also running the DNS service) once an hour. The registration server checks the log of the DNS queries, any source IP address matching the predefined sequence is taken as a potential MS. The dvping and additional conformance tests on registration server are then performed periodically with potential measurement server and the source IP address passed the test will be inserted into the database as the measurement server.

(2) Mapping
The mapping procedure described in Section2.2.3 is then used to assign ASN related domain names to the measurement server. If there are more than one servers exiting in the same ASN, the DNS load balancing techniques can be used to retrieve the IP address of the measurement server in a specific autonomous system. Since the DNS has been proven to be a very scalable scheme, it ensures the scalability of this registration process.

Figure 6 Auto-registration of new measurement server

2.3 SASM Framework
Figure 7 summarizes the general SASM framework. The dvicast, dvping and DVTS applications described in Section 2.1 working together do the application specific measurement test. The dvbeacon described in Section 2.2.2 ensures the performance inside an autonomous system. The dvtrace described in Section 2.2.4 finds the measurement servers along the packet traveling path and dvreg described in Section 2.2.5 provides a scheme for new measurement server registration. A newly installed measurement server will go through auto-registration process (dvreg) and listed in the database for the corresponding autonomous system. When user A and user B want to run DVTS application, they will do dvtrace to find the measurement servers along the packet traveling path and enable the measurement test (dvping, etc) with any discovered server at any time to find the network bottlenecks, then solve the problem and have a successful video session.

Figure 7 SASM framework

3. CASE STUDY
To demonstrate the effectiveness of SASM framework proposed in this paper, we have developed large-scale distributed measurement servers on national and international academic backbones CERNET2 (AS23910), CERNET (AS4538) and TEIN2 (AS 24489). Several experiments have been done among 25 Pops on CERNET2, among several Pops on CERNET and TEIN2 with DVTS-specific measurement tools installed to support for intra-domain and inter-AS DVTS application collaboration, respectively.
3.1 dvping, dvmcast and dvbeacon
Figure 8 (a) and 8 (b) show the measurement results from DVTS application and the dvping with dvmcast for a pair of nodes under good network condition. Both the end users and network administrators can understand each other concerning the video/audio quality and the network related parameters, such as throughput, the loss and the latency at the same time.

3.2 Measurement Server Auto-discovery
This test is held between a sender in AS24489 and the receiver is in AS 4538. The test topology, the traceroute result with IP address to ASN list mapping and ASN to the measurement server IP address mapping are shown in Figure 10, Figure 11 and Figure 12, respectively. It is clear that at least one measurement server can be obtained using the method proposed in SASM.
estimate the bandwidth resources availability for certain link. In addition, the scalable measurement server scheme can help the end-users find the test server effectively, especially for inter-autonomous system cases. With SASM, most of the high-performance video application problems could be solved prior to the real multi-sites rehearsal and the heavy negotiation load will be relieved.

The SASM framework can be easily extended for other real-time applications-specific measurement, especially for uncompressed HDTV video (1.2Gbps).

5. REFERENCES
ABSTRACT
Mobile phones have two sensors: a camera and a microphone. Our goal in this position paper is to explore the use of these sensors for building an audio-visual sensor network that exploits the deployed base of millions of mobile phones worldwide. Among the several salient features of such a sensor network, we focus on mobility. Mobility is advantageous since it yields significant advantage in spatial coverage. However, due to the uncontrolled nature of device motion, it is difficult to sample a required region with a given device. We propose a data centric abstraction to deal with this difficulty. Rather than treating the physical devices as our sensor nodes, we introduce a layer of static virtual sensor nodes corresponding to the sampled data locations. The virtual nodes corresponding to the required region to be sensed can be queried directly to obtain data samples for that region. We discuss how the locations of the virtual sensor nodes can be enhanced, and sometimes derived, using the visual data content itself. Experiments with real data are presented to expose some of the practical considerations for our design approach.

Categories and Subject Descriptors
C.2.4 [Computer Systems Organization]: Computer Communication Networks—Distributed Systems

General Terms
Algorithms, Design, Management

Keywords
mobile sensing, network organization, coverage, spatial resolution

1. INTRODUCTION
Mobile phones can be used as sensor nodes [1]. They all have a microphone, and most have a camera. Not only can the audiovisual data be processed to derive other interesting sensing modalities, but also additional sensors can be connected to a phone [18] using Bluetooth. Mobile phones are connected to a network infrastructure and have some form of data connectivity, ranging from short message service (SMS) capability to broadband wireless connectivity (eg. GPRS, 3G). The challenges in power management that affect many unattended sensor network deployments become relatively benign here, since a phone can be easily recharged in its user’s car, office, or home. These features make it feasible to use a phone as a sensor and thus we propose to build a sensor network using multiple individual owned cellular phones as its sensing substrate.

1.1 Sensing Advantages
Using mobile phones has several advantages for sensor networking applications. Firstly, a large number of cell-phones already exist around the world, providing the physical sensing infrastructure. Deploying the sensing hardware and providing it with network and power requires significant effort in other sensor networking systems.

Secondly, such a system can take advantage of the community effect. Many useful systems have been built where several contributors each develop and share a small component of a much larger system, such as Wikipedia, Linux, and other Web 2.0 applications. Such an approach, sometimes referred to as peer production [2], can leverage small amounts of sensor data contributed by mobile phone users to enable useful sensing applications. For example, a single mobile phone owner taking pictures of broken sidewalks on her fitness running route in an urban neighborhood may generate a dataset with very limited utility but if several such runners share their data, the utility grows significantly and the dataset may suddenly become useful for more activities such as damage mapping for city repair planning, new running route planning in runners’ e-groups, or urban lifestyle modeling for sports goods related business dashboards.

Thirdly, mobile phones can provide coverage where static sensors are hard to deploy and maintain. No single entity may have the access rights to place sensors across the complete coverage domain required by an application, such as a combination of subway stations, public parks, and shopping malls. The number of static sensors required to cover the same spatial expanse as covered by a single mobile device1 may be prohibitively expensive to deploy.

Further, each mobile device is associated with a human user, whose assistance can sometimes be used to enhance application functionality, such as by pointing the cellphone camera at the object of interest. Human assistance may be very limited, and may depend on application, but can often help overcome certain hurdles that are hard to overcome otherwise.

While the advantages of a shared system include the vast cov-

---

1There is a trade-off between spatial and temporal coverage when motion is used. Clearly, mobile phones may not cater to applications where continuous coverage in time is critical.
1.3 Key Contributions

Our overall project goals are to provide coordination and net-working mechanisms that allow multiple sensing applications to access third party shared resources in an efficient manner. Specifically, the first form of shared resources considered are mobile phones, due to their widespread availability. Building such a sensor network using uncoordinated mobile phones, where each phone is serving a different individual, the system as a whole is serving multiple sensing applications, and the phones move without any application’s control, involves many challenging issues. We discuss some of these challenges and present one approach to realizing a coherent shared sensor network based on this volatile swarm of mobile devices, in section 2.

Presenting the data collected by this highly volatile swarm in a usable manner to the sensor network applications makes it critical to obtain location information. Mobility helps increase the spatial coverage significantly, possibly by orders of magnitude, but without location, the samples taken by a mobile device cannot be associated with the corresponding spatial coverage. We discuss how location can be obtained at mobile devices using current technology. We also discuss how the audio-visual nature of the sensed data can be exploited to enhance location accuracy in section 3.

2. SYSTEM DESIGN

2.1 Assumptions and Requirements

We consider a sensor network of mobile phones which is built as a shared system. Each phone serves its local user’s needs first, such as making and receiving voice calls. The user continues to use any software, such as calendars or games, installed on their phone as they need. The sensor networking application only uses the phones as its sensors when available. Thus, the system can only work in best effort mode without any expectation of hard guarantees.

Our system does not control or even know the motion plan of the devices. Sensing requirements must be dynamically mapped to relevant devices in the underlying swarm of sensor devices.

Unlike dedicated and application specific sensor networks where most devices are homogeneous with known configurations, in the shared mobile phone sensor network, devices may be highly heterogeneous, not only in their hardware resources and bandwidth availability but also in terms of human user’s willingness to share. We require the sensor networking application to be able to accept the privacy and sharing policies set by the local user of each device.

Thus, different mobile phones may participate in the shared sensor network to varying degrees, based on their privacy sensitivity, willingness to share battery energy and bandwidth, device performance and capabilities, local workload, and willingness to provide human assistance for various applications.

2.2 System Architecture

In most current sensor network deployments, sensor nodes are accessed directly using their node ID’s or network addresses. While each phone has a unique network address (a phone number, among other unique identifiers) and can be accessed using that identity, this approach leads to difficulties in managing the spatial coverage due to node mobility. Also, this approach requires the application to know the network identities of the mobile phones in advance, which is a significant overhead for the application due to the large number of phones in the shared system.

We propose to use a data based abstraction that does not rely on the node identities of the physical devices. A layer of virtual sensor nodes is superimposed on the physical sensor network. A virtual sensor node at any point in space is based on data samples taken at that point. It is thus static. The stream of data coming from a virtual sensor node corresponds to data samples taken by a single or
multiple physical nodes when they visited that location. The stream may be sparse in time if that location is visited infrequently. Figure 1 illustrates this abstraction. The figure shows three key entities in the system. The first is the set of physical mobile devices that sense the world. The second is the virtual sensor layer. This exists only in the network infrastructure. The third are the applications that wish to use the sensor network of mobile phones.

All applications access the virtual sensor layer to obtain sensor data or to program sensing tasks. The physical nodes connect to the network infrastructure as convenient to upload data and may optionally download sensing commands submitted by applications. They may upload data in response to the sensing commands or simply based on what the device user wishes to share. The network infrastructure must populate the virtual nodes and the data streams of the virtual nodes based on the uploaded data. In our current prototype, the network infrastructure is a server hosting a web-service that allows the physical nodes to upload their sensed data (images, audio). The virtual nodes are all hosted by this server in the prototype using a database with data samples indexed by location. The location attribute serves as the virtual node address. While our prototype uses a single machine, in a production system, the virtual node hosting workload may be distributed among multiple servers, such as hierarchically organized by geographic regions, sensor modality or other criteria.

The virtual sensor layer can solve several problems in the shared mobile phone sensor network design. First, the addresses of the virtual nodes are naturally location based and the applications do not have to discover or manage the identities of the physical sensor nodes. The application may not care which specific phone captured an image as long as the required image is available to it. Of course, the physical device’s identity may be shared as an attribute associated with the data for applications that need it, when the device owner allows sharing their identity. Secondly, this removes the complications due to mobility, since the application no longer tracks the motion trajectories of physical mobile devices to determine which nodes to contact for its sensing needs. Thirdly, the physical device may not always be connected to the network, or may not wish to be contacted. For instance, the device may only upload its data at a convenient time, such as when it has spare bandwidth available. The virtual sensor layer makes this disconnected operation transparent to the applications as they always access the virtual nodes, which are available as long as the network infrastructure is available.

There are many other aspects of the system that present challenging and interesting design problems. A mechanism must be provided for the mobile phones to efficiently provide the collected sample data to the virtual sensor layer. The use of resources in communication and storage of data can be optimized by computing the value or relevance of the data for an application. The privacy of the mobile phone users and the sensed individuals should be respected and the data sharing methods must allow for the privacy sensitivity to vary across individuals and with the user’s spatio-temporal context. Since, the system is shared, methods are also needed to ensure the integrity of the data provided by unknown contributors. Another issue is the variability in the quality of data introduced by sensing with different phones passing through the same area. The data may need to be carefully calibrated with respect to each other and changes in data quality within the data stream of a single virtual node must be made known to the applications. A further challenge is to allow efficient programming mechanisms for multiple applications run concurrently. Mechanisms are also needed to allow applications to seek human assistance in their data collection processes. In our prototype, a data publishing client has been developed for mobile phones that uses the server’s web service to upload its sensed audio-visual data. The client can be configured to collect data automatically or only when the human user explicitly triggers a data sample capture, such as taking a picture. Mechanisms to program the data collection activity in response to application needs is part of our ongoing work. A prototype of our system that addresses some of these challenges appears in [9]. There are many other possible ways to architect the shared mobile phone sensor network and many design variations even within the above architecture, that are not discussed here. The focus of the remainder of this paper is on obtaining location information for making the volatile sensor layer easy to access for applications.

3. LOCATION

To realize the virtual sensor layer proposed above, a key requirement is that the data samples collected by a phone be location stamped. There are many methods to obtain location on a mobile phone:

1. **Cell-tower triangulation**: The cell-phone network typically knows the location of a phone using signal strength measurements at a phone from one or more cellular base stations. This location is accurate to several meters when the phone is in a region with three or more base stations in range. The location accuracy falls to the granularity of a mile or more when only one base station is in range, such as in rural areas. This location information can be accessed using commercial products such as the Mappoint Location Server [11] when the phone is connected to a cellular network that does make this information visible to non-operator owned applications, such as Sprint in the US, TeliaSonera in Europe, or Bell Mobility in Canada.

2. **Phone GPS**: Many recently released and forthcoming mobile phones have built in GPS receivers and these phones can know their location using the GPS system. The location information is accurate to several meters when the phone has good GPS satellite visibility, such as under open sky. The location is typically not available when indoors.

3. **Wireless LAN triangulation**: Many mobile phones, especially newer models, have built in wireless LAN capability. Location can be obtained using WLAN signal strength based triangulation when multiple access points with known location are in range achieving accuracy of under a meter, or sim-

![Figure 1: A block diagram of the shared sensor system.](image-url)
...by the location of the access point when only one is in range, achieving an accuracy of several meters. When the location of the access point itself is not known, the location may be estimated from the external IP address for the access point. These techniques are accessible on WLAN connected devices through already available services such as the “Locate Me” feature on LocalLive.com.

4. Human entered tags: The phone user may key in tags for all or some of the images taken by her. Some of these tags may include address or landmark information that helps infer the location.

3.1 Content Match and Location

A combination of the above techniques may be applied in practice to location-stamp the samples collected by a phone. However, all these methods have limitations in terms of accuracy and availability. One method to help overcome some of these limitations is to exploit the visual data content itself to enhance the location accuracy for sample points and also to obtain location in scenarios where the above methods are unavailable. The underlying assumption in exploiting the visual data is that two or more images taken at the same location are likely to have some common visual component. This enables the following location enhancing alternatives. For images that have no or highly inaccurate location, such as from human entered landmark tags or from the registered home address for a mobile device, matching the content of the image against other images with known locations can help assign a more accurate location stamp to the image. In situations where the location data is somewhat inaccurate, such as indoor locations when GPS is used, or rural locations when cell-tower triangulation is used, content-based matching can help place related images together. For instance, for all images taken within a building, such techniques may help differentiate among images taken in different rooms. We explore the use of visual data content for location enhancement in the latter scenario in more detail.

Note that while content matching helps assign location to visual data alone, correlating the location with the time-stamp of the image also tells us the location of the mobile device at the time the image was taken. Thus, all other data samples from the same device, such as audio data or samples from other sensors connected to the phone, taken with nearby time-stamps, can be assigned a location-stamp as well. Suppose a large number of images is contributed by several mobile phones participating in the shared sensor network. Suppose next that these images have been separated into multiple virtual nodes based on the location stamps, where images with location stamps within a small threshold distance \( \delta \) of each other are assigned the same virtual node.

Consider a virtual node corresponding to several images taken within a building. Suppose that mobile phones that used GPS location assigned the last known GPS location, obtained just before GPS satellite visibility was lost - the location of the building entrance. Then, all these images corresponding to different floors and rooms within the building would be assigned to the data stream of this single virtual node. Also included in the same stream would be images that had location stamps within \( \delta \) of this virtual node’s location but in fact correspond to other locations, such as an image taken a mile or more away but assigned the same location due to cell-tower localization error. Our goal is to show that content matching can be a useful technique to cluster together images that belong to the same location (such as a room) within the building and separate out images assigned to this virtual node in error.

3.2 Algorithms

We use the term zones to refer to the finer granularity regions within a common geographic vicinity. For instance, different shops in a mall, or different aisles in a store, may be termed as different zones. The problem of refining location based on content for a set of images assigned the same locations by the applicable location technology may be broken up into three parts.

First, for each new image that is assigned to this geographic vicinity, we need to determine which zone within that vicinity it belongs to. If the zones are known, then, a suitable matching technique is needed to select the most closely matching zone for the image, and to reject false matches. Each zone may contain multiple existing images, and the new image may yield matches with multiple images in different zones.

Second, the zones themselves may not be known a priori, and may be required to be determined from the image data itself. This is a hard problem since neither the number of zones, nor a distribution of images among those zones may be known.

Third, the zones may have to be associated with geographic location. Matching the images by content and separating them into matching zones does not itself yield information about which zone is located in which physical area of the geographic vicinity.

There are several possible methods to compare the content of image data, such as color histograms, texture matching, and key feature matching. However, none of the content matching techniques is perfect and yield a non-negligible number of false matches. The false match problem is acute in our problem setting since all images being compared do belong to the same geographic vicinity. For instance, images taken in different rooms of a building are likely to contain some similar content. We select key feature matching [10] due to its robustness to lighting changes, image size variation, and imaging device changes. Key features correspond to selected objects in a scene such as corners or peculiar textures that are expected to be preserved across images taken from different points of view and in different lighting. Key features in two images that correspond to the same physical object are likely to match. Fig 2 shows two images with matching key features. It may be seen that while some matches correspond to the same physical objects, there are false matches also.

![Figure 2: Key features for content matching.](image-url)
We use the following procedures to exploit key feature based content matching for location assignment. Suppose $n_i$ represents the number of key features in an image $i$ and $n_{ij}$ represents the number of matching key features across two images $i$ and $j$. We define a matching metric $m(i, j)$ between two images based on their key features as:

$$m(i, j) = \frac{n_{ij}}{n_i + n_j} \quad \text{if} \quad i \neq j$$

(1)

and $m(i, i) = 0, \forall i$.

**Zone Discovery Algorithm:** Suppose $N$ images have been assigned to a geographic vicinity and we wish to separate these into $K$ zones where $K$ is unknown.

Consider a graph $G$ with $N$ vertices each corresponding to an image and weighted edges between them, where the weight of the edge between vertices $v_i$ and $v_j$ is $m_{ij}$. Define a binary relation $R$, that relates each node to its neighbor connected with the maximum weighted edge:

$$R(v_i, v_j) = \left\{ v_j \mid \max_{j \in \{1, \ldots, N\}} m_{ij} \right\}$$

(2)

subject to $m_{ij}$ being higher than a threshold, $m_0$. We cluster together images that belong to the same zone by computing the transitive closure, $R^\ast$ of this relationship for all vertices. The algorithm starts computing a path from a vertex $v_i$ to its neighbor with the maximum weighted connecting edge, until, another node in the same path or in a previously assigned sub-graph (path generated from a previously processed node) is reached. This separates $G$ into multiple sub-graphs, $g(k)$, each corresponding to a zone. This is a conservative procedure in the sense that images assigned the same zone are likely to belong to the same physical zone, but multiple zones may have been generated for a single physical zone, such as when there is insufficient overlap among images from two different corners of a room. Other possibilities such as based on unsupervised learning, using small training sets, or when $K$ can be determined using alternate means are also of interest, and are part of our ongoing work.

As sensors feed in more data samples, each new image must be assigned to one of the previously determined zones. While it is possible to re-compute the zones each time a new image is obtained, that has a high computational overhead. Instead, the new image may be assigned to one of the existing zones based on its content match. Prior work has considered this problem [17, 7] when an image data-set with known locations is available or constructed from a dense corpus of overlapping images, and test images are compared against this set to determine the location of the test image. Further scalability concerns have been addressed in [12, 15] for matching against large number of images. These techniques are applicable to our problem scenario. However, the problem is simplified in our scenario since rather than comparing a test image against the entire corpus of previously known images, we only need to compare it within a small number of zones of a geographic vicinity and the number of images representing each zone is likely to be small.

As an illustration, consider using key feature based content match. Suppose a small number of images $n_k$ has been assigned to a zone $k$, where $k = \{1, \ldots, K\}$. We use the following metric to match a new image $i$ against multiple images in zone $k$:

$$M_k(i) = \frac{1}{n_k} \sum_{j=1}^{n_k} m(k, j)$$

(3)

where $m(k, j)$ is the normalized key feature based match score as defined in (1). The zone assigned to the image is simply the $k$ that yields maximum $M_k(i)$.

The zone assignment process can also be iteratively used to refine the zones determined previously, such as when many images with two or more competing zone matches are found, it may lead to merging the separate competing zones into a single zone. It is worth noting that the binary relation defined in (2) may be asymmetric and this can indeed lead to such merging depending on the order in which the images arrive.

Each zone also needs to be associated with a physical location. This could be achieved based on some of the images for which fine grained location is known, such as through human supervision, opportunistic GPS signal received near windows, or when the number of images is sufficiently large, using geometric 3D reconstruction from the images themselves.

4. EXPERIMENTS

In this section we illustrate some of the algorithms proposed above on real image data. We collected 150 images within a building, with a majority corresponding to two distinct zones: a conference room and a kitchen lounge, and the rest arbitrarily distributed across building corridors and an individual office on the same floor. All these images represent image samples contributed by mobile devices that use GPS location and as the GPS signal is unavailable indoors, all these images are assigned the same GPS location - that of the building entrance. Our goal is to explore content matching methods to separate these images into distinct zones within the building.

We first computed the key features and the inter-image match as defined in (1) for all image pairs. Then, we applied algorithm 1 to separate the images into distinct zones. Also, to enable us to measure the algorithm’s error, we manually recorded the true zone for each image. The error is defined as follows: for each assigned zone, we check whether all the images belong to the same true physical zone. The true physical zone of an assigned zone is taken to be the physical zone corresponding to the majority of the images for that assigned zone. Images which do not belong to the corresponding physical zone are counted as erroneous images and dividing this count by the total number of images assigned to the zone yields a normalized error for each assigned zone.

![Figure 3: Distribution of zone errors.](image)

We obtained an aggregate error of 12.52% averaged over all zones. However, the error has a high variation, with some zones showing zero error and others showing a very high error. A histogram of the error across 16 zones is shown in Figure 3. It may be seen that while most zones have a very low error, there are some zones with very high error. Manually examining the data
corresponding to those zones revealed that these are zones which correspond to physical zones with small number of images. This is understandable as key feature matching is likely to yield fewer true matches in these cases making it more likely for the maximum weight neighbor to be a false match.

Another parameter determined by the algorithm is the number of zones. As mentioned, the algorithm is conservative in combining images into the same zone, causing it to overestimate the number of zones. Multiple zones are generated for the same physical zone when there is insufficient overlap among images. In our dataset, we obtained 16 zones, as opposed to five true zones.

The second aspect illustrated experimentally is the assignment of zones to images when the zones themselves are known. In our data-set, the first 100 images correspond to the two zones with majority of the images. We used 45 of these as a base set which has been divided into their true zones. The remaining images were assigned to the two zones using the metric in (3). Comparing the results with the ground truth, yields 20.01% error, where an error implies that the image was assigned to the wrong zone. More conservative zone assignment methods may be used to reduce the error, at the possible expense of leaving some images unassigned. As the number of images grows, the error will decrease since more overlap and matching regions will exist among images. The results also indicate the limitations of the specific key features used in this implementation for computing image matches, and suggest the need for exploring other content matching techniques to realize our data centric abstraction.

5. RELATED WORK

Several projects have considered the use of mobile devices for building sensor networks. Specifically, the use of mobile phones has been proposed in [16, 13]. Large scale sensor networks using mobile devices dedicated to sensing, and carried by people or vehicles, have also been proposed [1, 8, 4, 3, 14]. Our techniques of using location based virtual sensor nodes rather than application level tracking of the highly mobile physical devices is relevant to all these projects. The use of visual data content for location enhancement is applicable as well, when the mobile sensors have a camera as part of their sensor suites.

The use of similarity among multiple data samples, specifically text documents and images, has also been considered in [6] for data collected by a single individual. Our goal is to leverage data similarities in sensor data from multiple individuals to provide a location based abstraction. Sharing of sensor data is already available at earthcam.com for sharing network camera feeds, on weatherunderground.com for posting weather sensor data, and sensorbase.org for some other sensors. These applications demonstrate the viability of sharing sensor data and also motivate the need for methods that facilitate sharing, such as the ones presented in this paper.

6. CONCLUSIONS

We proposed to use the large number of mobile devices present in our environment as a sensor network. Among many interesting challenges in realizing this vision, we focused on using location information to make this highly volatile and mobile swarm of sensor devices usable by sensor networking application without having to track the device motion trajectories. In addition to using many of the available location technologies, we showed that for audio-visual sensor data, the data content itself can be exploited to refine location granularity. Even when the physical location cannot be determined, clustering together images which correspond to the same zone within a larger geographic vicinity can be advantageous for application interested in sensing a specific zone. While the content based techniques are directly applicable to visual data, correlating the time-stamps and device identities across samples can allow the network infrastructure to organize samples from other sensing modalities into location based virtual nodes as well. We have presented some of our very initial experience in organizing a mobile device based sensor network. This domain presents several interesting research problems, and the audio-visual nature of the sensed data can be exploited in solving some of them. Future work includes exploring sophisticated content matching, such as based on combination of multiple image features, the use of motion trajectories learned from motion patterns, and correlation of other sensor data in inferring user location.

7. REFERENCES

QoS Adaptation for Realizing Interaction between Virtual and Real Worlds in Pervasive Network Environment

Shinya Yamamoto, Yoshihiro Murata, Naoki Shibata\textsuperscript{1}, Keiichi Yasumoto and Minoru Ito
Graduate School of Information Science, Nara Institute of Science and Technology
Ikoma, Nara 630-0192, Japan
\textsuperscript{1} Department of Information Processing and Management, Shiga University
(shiny-ya,yoshihi-m,yasumoto,ito)@is.naist.jp, shibata@biwako.shiga-u.ac.jp

ABSTRACT
We propose a framework called FAIRVIEW which realizes cooperative work and interaction between mobile users in real world and remote network users. FAIRVIEW allows mobile users and network users to share the same view of a shared space including many moving objects in sufficient quality for interaction. For this purpose, we devised a mechanism which continuously measures information (called AR information) of the position and direction of each object and delivers AR information to user terminals, so that users can see moving objects in real-time. In order to realize real-time delivery of AR information with limited network resource in an ordinary wireless LAN and Internet environment, we propose a QoS adaptation mechanism which allows users to observe more important objects with a higher framerate. Through experiments supposing network environments with various available bandwidths, we confirmed that the proposed mechanism achieves a practical framerate of moving objects between mobile users and remote network users in pervasive network environment.

Categories and Subject Descriptors
C.2.4 [Distributed Systems]: Distributed applications

General Terms
algorithms

Keywords
Networked Virtual Environment, Interaction, QoS Adaptation, Augmented Reality

1. INTRODUCTION
Recently, there are many studies regarding to MR (Mixed Reality) and AR (Augmented Reality)[6, 8, 18]. Many research efforts have also been made for NVE (Networked Virtual Environment) and CSCW (Computer Supported Cooperative Work) [7, 17].

\textsuperscript{*}This work was partly supported by Hayao Nakayama Foundation.

These technologies allow remote users to participate in social activities such as shopping, exhibition, sports, and game which are held in real space. For realizing DVE (Distributed Virtual Environment), the following five criteria should be satisfied: (1) real and virtual users share a common virtual space; (2) users can freely change their positions and directions, and the changes are instantly reflected in other users’ views; (3) each user can introduce objects into the shared space, and make actions, such as pushing and holding objects, and reaction should be reflected in other users’ views; (4) the required apparatuses should not be special nor expensive; and (5) a massive number of objects can exist in the shared space. There are some studies on communication architectures for MMOG [1, 4, 13] and NVEs for remote cooperation [11, 12]. Some of these existing studies realize scalability on sharing virtual space between many users using P2P technologies. However, they allow sharing virtual space and objects among only virtual users. On the other hand, the existing MR and AR technologies, the criteria (1) to (3) can be satisfied. However, they require special devices, servers and networks, and thus satisfying the criteria (4) and (5) is difficult.

In this paper, we propose a framework named FAIRVIEW which realizes smooth cooperation and interaction between real and virtual users satisfying all the criteria (1) to (5) using inexpensive devices off the shelf. To satisfy the criteria (1) to (3), FAIRVIEW produces a hybrid shared space by overlapping a virtual space and a real space, and provides a mechanism for allowing the virtual and real users to observe each other. To satisfy the criterion (4), we suppose that virtual users use ordinary PCs with an internet connection, and that real users use wearable computers with HMDs (head mount display) or PDAs, with internet connection via wireless LAN. In FAIRVIEW, the information regarding to orientations and positions of real objects (called AR information, hereafter) are measured at short intervals using an existing AR measurement tool. The information is exchanged among user terminals, and the object is displayed as a 3D graphics on the display of virtual user terminal. To achieve the criterion (5), we propose a mechanism for delivering AR information as well as action/reaction to object in real time (which we call AR event delivery mechanism, hereafter) to realize smooth cooperation among real and virtual users. AR event delivery mechanism includes a QoS adaptation mechanism for controlling the intervals of transferring AR events between users so that the total transmission rates will not exceed the limit of available bandwidth. The adaptation mechanism decides the transfer intervals according to importance of each object for each user, which is determined automatically according to the distance and position of the object in the user’s view.

To evaluate the proposed mechanism, we analyzed the required bandwidths and investigated the effectiveness of our QoS adaptation mechanism under some configurations with different numbers...
of objects. As a result, we confirmed that the proposed mechanism realizes smooth interaction involving a large number of real and virtual users/objects on an ordinary wireless LAN and internet environment.

2. RELATED WORK

In [1, 4, 13], efficient AOI (Area of Interest) management methods are proposed. In [13], game space is dynamically divided based on Voronoi diagram for direct communication among players in a same fragment. In [4], the space is divided into small areas called micro cells. To distribute the load for processing events among multiple servers, the regions managed by each server are dynamically changed. In [1], the shared virtual space is divided into honeycomb regions, and a mechanism based on Pastry[2] is used to allow each user to receive information for players and objects in the player’s AOI.

Although these existing NVEs enable efficient information exchange among virtual users, it is difficult to apply them to the mixed space of real and virtual worlds which requires a large amount of information exchange in real time on resource-limited wireless network, which is the target environment of FAIRVIEW.

In [11, 12], a load distribution and a QoS adaptation mechanisms for DVEs are proposed, respectively. In [12], in order to cope with so called area boundary problem (inconsistency caused by neighboring areas managed by different servers), the whole shared space is managed by each of game servers, and game processing tasks are flexibly distributed among the servers. Unfortunately, this method supposes high performance servers and high-speed networks and treats only virtual users. In [11], an IPv6-based network architecture called VESIR-6 is proposed for realizing a large-scale DVE where users can share a 3D virtual space and objects. Aiming at efficient utilization of network resources, VESIR-6 uses multicast for delivering object state updates to users, anycast for load distribution among servers, and IntServ/DiffServ-based QoS adaptation mechanism for regulating per-flow transmission rate. However, VESIR-6 does not suppose wireless network environment which is necessary for interaction between real and virtual users. Also, delivery of object state updates is managed only by joining/leaving the corresponding multicast group. Therefore, VESIR-6 cannot provide fine-grain QoS adaptation like the proposed method.

The most related study to our work is tele-immersion which captures the whole environment and reproduces it at geographically distant location. TEEVE[14] displays 3D live visuals from each user’s view in real-time, and constructs an environment for cooperative working. However, expensive and specialized devices and infrastructures such as 3D cameras and broadband network environments are needed to construct tele-immersion environment.

3. OVERVIEW OF FAIRVIEW

This section overviews the functions of FAIRVIEW and presents example applications, then describes the target environment and the basic ideas for implementation.

3.1 Functions of FAIRVIEW

FAIRVIEW overlaps real space and virtual spaces, and provides the environment where users in real space (called real users) and users in virtual spaces (called virtual users) can interact as if they were in the same space. We call the overlapped space hybrid space.1 The main functions of FAIRVIEW are as follows: Function (1) providing the same view to real/virtual users: Virtual users can change their positions in virtual space using keyboards and mice like ordinary 3D first person shooting games. Real users can ordinarily move using their feet in real space. The users see objects in both virtual and real spaces according to their positions/directions. Function (2) Voice conversation among users: Users can talk with each other using their voice. The user’s voice can be heard according to the user’s position. This function can be realized with the technique in [16]. Function (3) Object sharing: Real objects and virtual objects can be registered in (and also unregistered from) the hybrid space. The registered objects can be seen by both virtual and real users. Users themselves are also objects.2 Shared objects can be seen by all users whose views include the objects. We assume that 3D geometry data for the registered objects are prepared beforehand. Function (4) Moving and bonding objects: Users can move neighboring objects. Virtual objects can be moved by both real and virtual users, while real objects can be moved only by real users. Movement of objects can be observed by other users. Two or more objects can be bonded by specifying their relative positions. Bonded objects basically move together, but if a user moves a virtual part of bonded objects made of real and virtual objects, the bonded object is disengaged.

3.2 Applications of FAIRVIEW

We describe two example applications which are enriched by using the functions of FAIRVIEW described above.

Virtual flea market  In this application, real and virtual users trade goods in a real market. Virtual users go shopping to a flea market in hybrid space. Real sellers register their goods to hybrid space. Virtual buyers see what kinds of goods are sold from distance. The sellers and buyers interact with each other via gestures and voice. The sellers show and explain details of their goods to virtual users by rotating and moving the goods. This kind of application includes exhibition, trade fair, and shopping center.

Multiplayer game  In this application, virtual users participate in a paintball wargame held in real space. In paintball wargames, players possess airguns and targets, and shoot opponent groups’ targets. Players whose targets are shot lose the game. Real players only need to register airgun and target. The virtual users cooperate with other real or virtual users. Virtual characters like a huge dinosaur can participate in the game. Players can be added in the case when there are too few real users. This kind of application includes attractions in theme parks, events on a street corner.

3.3 Devices and Network used in FAIRVIEW

Table 1 shows the list of necessary equipments for the users. Real users use wireless small computing devices such as PDA, and HMD space and virtual space.

---

1FAIRVIEW is also capable of overlapping more than one distant real spaces and provides the same view for the users in those spaces. For simplicity, we focus on interaction between the users in real

2Unregistered real objects can also be seen by real users, but in this paper we assume that all objects are registered.
with which the real view can be seen as the background of virtual view. These devices should not hinder real users’ movements. The equipments also include positioning devices such as GPS receivers, sensors to detect position and direction, and audio input/output devices. We assume that inexpensive devices are available for these purposes. We use webcams and dedicated softwares like ARToolkit [18] for positioning and detecting orientations of objects. Virtual users use ordinary PCs for FAIRVIEW.

3.4 Basic Ideas to Implement FAIRVIEW

To realize an application using AR technology, we considered how the views seen by users are rendered inexpensively. In TEEVE [14], images captured by 3D multi-cameras are processed and transferred through Internet2. Since we are aiming at inexpensively implementing the functions, we decided to measure AR information using inexpensive sensors. To realize this method, we have to resolve the following three problems: (i) measuring AR information accurately; (ii) transferring the measured AR information in real time; and (iii) rendering the objects.

For resolving the problem (i), we use GPS receiver if a user is outside of building. If a user is inside building, we use an indoor positioning method such as a method using wireless LAN access points [9], a method using speakers and microphones [5], Place Lab [19], and Weary [10]. The orientation of an object can be measured using rotational and translational acceleration sensors, methods based on image processing such as ARToolkit [18] or the method in [6]. For the problem (iii), we assume that users have terminals capable of rendering 3D graphics. The problem (ii) is the main problem which we treat in this paper. To resolve this problem, we need a delivery mechanism with which AR information can be exchanged among wireless and wired users in real time. We call this mechanism AR event delivery mechanism. In applications such as flea market or paintball wargame, hybrid space may have terminals and bandwidth shortage can occur as the amount of such AR information becomes large. Therefore, we have to measure the available bandwidth between real user terminals and AP. Note that all real user terminals share the bandwidth. For each \( vn \) in \( VN \), let \( bw_{VN}(vn) \) be the available bandwidth between \( vn \) and \( AP \). For each \( vn \) in \( VN \) and \( rn \) in \( RN \), let \( bw(vn, rn) \) be the available bandwidth between \( vn \) and \( rn \). Note that \( bw(vn, rn) = \min(bw_{VN}(vn), bw_{VN}(rn)) \). For each pair of virtual user nodes \( vn1 \), \( vn2 \) in \( VN \), let \( bw(vn1, vn2) \) be the available bandwidth between \( vn1 \) and \( vn2 \).

4.4 Assumption on User Communication

We suppose that the whole real space \( R \) is covered by only one AP connected to the Internet. Therefore, each real user terminal in \( RN \) can communicate with any virtual user terminal in \( VN \).

Let \( BW_{AP} \) be the available bandwidth between real user terminals and AP. Note that all real user terminals share the bandwidth. For each \( vn \) in \( VN \), let \( bw_{VN}(vn) \) be the available bandwidth between \( vn \) and \( AP \). For each \( vn \) in \( VN \) and \( rn \) in \( RN \), let \( bw(vn, rn) \) be the available bandwidth between \( vn \) and \( rn \). Note that \( bw(vn, rn) = \min(bw_{VN}(vn), bw_{VN}(rn)) \). For each pair of virtual user nodes \( vn1 \), \( vn2 \) in \( VN \), let \( bw(vn1, vn2) \) be the available bandwidth between \( vn1 \) and \( vn2 \).

4.3 AR Event Delivery Mechanism

We choose some of the user terminals as server nodes to manage efficient AR information exchange. The data including AR information exchanged among user nodes is called AR event, hereafter. In order to reduce processing and traffic load of each server node based on user’s AOI, we divide the whole shared space \( H \) into small rectangular sub-areas, as shown in Fig. 2, and assign a server node called area node to each sub-area. This is a similar approach to existing P2P-based MMOG gaming architectures such as [15]. Let \( an_A \) be the area node assigned to sub-area \( A \).

The sub-area \( an_A \) receives AR events of objects in \( A \), and delivers the AR events to users (in \( A \) and neighboring sub-areas) who are watching the objects.

In FAIRVIEW, since the network resource is tight due to wireless communication and calculating the reaction of an object as a consequence of an action is heavy for a real user terminal, we introduce two types of new server nodes called object management node (om-node) and bandwidth controller node (bwc-node).

One om-node is prepared for each virtual object except for virtual users’. Let \( on(o) \) be the om-node for virtual object \( o \) in \( VU = VU \). Then \( on(o) \) produces the reaction of \( o \) when a user or an object takes an action (e.g., pushing, holding) to \( o \). When \( on(o) \) receives an action event from a user node, \( on(o) \) calculates \( o \)'s reaction with physical simulation, and delivers a series of \( o \)'s AR events to users who are observing \( o \).

One bwc-node is prepared for each virtual user terminal \( vn \) in

---

**Table 1: User Equipment**

<table>
<thead>
<tr>
<th>Type</th>
<th>Computer</th>
<th>Display</th>
<th>Network</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real User</td>
<td>PC/PDA</td>
<td>LCD/ltc</td>
<td>WiFi/ltc</td>
<td>Sensor/Weam</td>
</tr>
<tr>
<td>Virtual User</td>
<td>PC</td>
<td>LCD/ltc</td>
<td>Internet</td>
<td>Mouse/ltc</td>
</tr>
</tbody>
</table>

---

\( vvec \) be the horizontal standard vector of the vertical standard vector of \( H \) terminal of produced by overlapping \( RH \).

4.1 Notation

Let \( R \) and \( V \) be the target real space and the corresponding virtual space, respectively. Let \( H = (R, V) \) be the hybrid space produced by overlapping \( R \) and \( V \). We suppose that \( H \) is an axis-aligned rectangle on \( x-y \)-plane of 3D coordinate system. Let \( hvec = (1, 0, 0) \) and \( vvec = (0, 0, 1) \) be the horizontal standard vector and the vertical standard vector of \( H \), respectively.

Let \( RO = \{r01, ..., r0m\} \), \( VO = \{v01, ..., v0n\} \), \( RU = \{ru1, ..., ruu\} \) and \( VU = \{vu1, ..., vuw\} \) be the set of real objects in \( R \), the set of virtual objects in \( V \), the set of real users (i.e., mobile users) and the set of virtual users (i.e., remote PC users), respectively.

Note that \( RU \subseteq RO \) and \( VU \subseteq VO \). Let \( node(u) \) be the user terminal of \( u \) for each user \( u \in RU \cup VU \). \( RN = \{node(u)|u \in RU\} \) and \( VN = \{node(u)|u \in VU\} \) are the real user terminals and the virtual user terminals, respectively.

For each \( o \) in \( RO \cup VO \), let \( AR(o) = (pos, angleH, angleV) \) be the AR information of \( o \), where \( pos, angleH, \) and \( angleV \) are the position on or beyond \( H \), the horizontal angle to \( hvec \), and the vertical angle to \( vvec \), of object \( o \), respectively. Each item of AR information is referred to by \( e.g., o.pos \).

We suppose that each real user terminal in \( RN \) can measure its user’s AR information at 60 times per second with equipment explained in Sect. 3. We also suppose that AR information of each real object \( ro \) except for users can be measured by a user terminal in \( ro \)'s proximity with ARToolkit and webcam.
nodes, and om-nodes are omitted due to the space limitation. We all real users to watching each virtual object at the same framerate.

For (2), node(u) sends the information as AR event to the area node an_A, and draws the latest appearance of objects on display of User Node. Consequently, we construct the overlay network per sub-area, consisting of an area node, N object nodes, M user nodes, and M bwc nodes, as shown in Fig. 3, where N and M are the numbers of objects and users in the subarea, respectively. We will explain how these nodes exchange AR events below.

User Node In FAIRVIEW, (1) a user u in RU ∪ VU can watch other objects in its view, (2) u can be watched by other users since u is also an object, and (3) u can take an action to other objects.

For (1), node(u) measures u’s AR information continuously and if the information differs from the last measurement, node(u) sends the information as AR event to the area node an_A (Fig. 3 (a)). For (2), node(u) receives the AR events of the objects in u’s view and draws the latest appearance of objects on display of node(u). To receive AR event, node(u) uses publish/subscribe model [3]. Once node(u) sends its AR event to the area node an_A, an_A automatically identifies the objects in u’s view, and forwards the AR events of the objects to node(u) via the bwc-node (Fig. 3 (b)). As shown in Fig. 3 (b’), the real user terminals receive the AR events from the bwc-node assigned for real space R via wireless AP by broadcast. For (3), when u takes an action to virtual object o, node(u) sends an action event containing power and direction, to an_A (Fig. 3 (c)). Then an_A forwards the event to o’s om-node on(o), and on(o) calculates the reaction and sends the AR events via the bwc-node to the users who are observing o (Fig. 3 (d)).

Area Node For each sub-area A of the hybrid space H, a virtual user node is selected and assigned to the area node. A virtual node with sufficient network and computation resources is selected, e.g., by the lobby server when the application starts.

The area node an_A manages the positions of the objects as well as the users’ view in the sub-area A. When an_A receives the AR events of all objects in A from the corresponding user nodes and object nodes (Fig. 3 (c)), an_A identifies the users who are watching part of other sub-areas neighboring A, based on their views calculated from their AR events (e.g., pos and angleH), and sends the AR events to the neighboring area nodes if needed. The sub-area an_A also receives the AR events of such users in other sub-areas from the neighboring area nodes. Finally, an_A sends the AR events of the objects in A to the user nodes via the corresponding bwc-nodes.

Object Management Node The om-node on(o) manages the reaction to o. One node is assigned to each object, but one node may manage multiple objects.

For the action taken to virtual object o, on(o) calculates the reaction with physical simulation and sends AR events for the reaction to o’s watcher nodes via the corresponding bwc nodes. Bandwidth Controller Node One bwc-node is assigned to each user node, although one node may serve as the bwc-nodes of multiple users. The bwc-node bn(u) monitors available bandwidth to its associated user node node(u), and regulates transmission rates of AR event streams.

5. VIEW-ORIENTED QOS ADAPTATION

The basic ideas of our QoS adaptation mechanism are as follows: (1) we decide relative importance value of each object according to how important the object is for a user; and (2) for each user, we regulate transmission rates of AR events of observable objects based on their importance values so that the sum of transmission rates are less than the available bandwidth.

5.1 Decision of Importance Value

Let Watcher(o) be the set of users who can observe an object o in RO ∪ VO. The set of users Watcher(o) is defined by

\[
\text{Watcher}(o) = \{u \in RO \cup VO, View(u) \subseteq o.pos\} \cup \{u \in RU, View(u) \subseteq o.pos\}
\]

Where View(u) is u’s view on hybrid space H, and represented by a half circle as shown in Fig. 4.

For each object o in RO ∪ VO and each user u in Watcher(o), let Imp(o, u) be the importance value of o for u. Like in the real world, the importance value Imp(o, u) should increase as the distance between o and u is shorter and o is located nearer the center of u’s view. Therefore, we define each user’s view as a half circle and divide it into 15 zones with five levels of importance a, b, c, d and e, as shown in Fig. 4. Here, the objects located on zone a have the highest importance, and the importance of objects on zones b, c, d and e decreases in this order.

5.2 Decision of Frame rates of AR Events

The transmission rate of AR events for each object towards user u is decided based on the ratio of its importance value to the sum of importance values of all objects in u’s view. The available bandwidth is distributed among the objects, and the framerate of AR events for each object is decided from the assigned transmission rate and the size of each AR event.

As mentioned in Sect. 4, AR events of each object are delivered to each user node through the corresponding bwc-node (see Fig. 3). The bwc-node drops packets of the AR events so that the transmission rate does not exceed the assigned bandwidth.

We describe our view-oriented QoS adaptation mechanism using examples. The QoS adaptation for AR event streams differs depending on the receiver type (i.e., virtual user or real user). Thus, we give two examples in the following subsections.
5.2.1 QoS Adaptation for Virtual User

Suppose that a virtual user $v$ in $VU$ is watching three objects $o_1$, $o_2$, and $o_3$ in his view. In this case, $\text{node}(v)$ receives the AR events of those three objects via the bwc-node $bn_v$ in $VN$ (see Fig. 3).

We assume that the available bandwidth between $bn_v$ and $\text{node}(v)$ is 1Mbps. We also assume that transmission rates for AR event streams of objects $o_1$, $o_2$, and $o_3$ are 0.5 Mbps respectively, that is, the sum of the streams is 1.5 Mbps. In this case, the available bandwidth (1 Mbps) is distributed according to the importance values of objects $o_1$, $o_2$, and $o_3$. Suppose that the importance values of objects $o_1$, $o_2$, and $o_3$ are 10, 25, and 15, respectively. As a result, portions of bandwidth $10 \times 10Mbps = 2Mbps$, $25 \times 10Mbps = 5Mbps$, and $15 \times 10Mbps = 3Mbps$ are assigned to the AR event streams, respectively. Based on this result, the bwc-node $bn_v$ controls transmission of the AR event stream of each object by dropping some of the received packets.

5.2.2 QoS Adaptation for Real User

Suppose that a real user $r$ in $RU$ is watching three virtual objects $o_1$, $o_2$, and $o_3$ in his view. In this case, $\text{node}(r)$ receives the AR events of those three objects via wireless AP and the bwc-node $bn_r$ in $VN$ (see Fig. 3).

In this case, we set the importance value $\text{Imp}_o(o)$ of virtual object $o$ in VO observed by real users to the maximum value among the users observing this object. That is, we define that $\text{Imp}_o(o) = \max_{\text{Watcher}(o)}(\text{Imp}_o(u))$.

The framerate of AR events of each object is decided similarly to the case in Sect. 5.2.1, and the bwc node $bn_r$ applies the QoS adaptation to the AR event streams to real users.

6. EXPERIMENTS

In order to evaluate the proposed method, we measured the required bandwidth for application of users and objects. We also measured the framerates at which users can watch the objects for two cases with and without our QoS adaptation mechanism. We also evaluated user satisfaction by questionnaire.

6.1 Configurations

Experimental configurations are as follows. The sizes of virtual space $V$ and real space $R$ are both $50 \times 50$ m. There is one wireless AP whose radio range covers the whole real space $R$. The available wireless bandwidth $BW_{AP}$ is 10Mbps, and all real users share this bandwidth. One area node and bwc-nodes are allocated on PCs on fixed wired network. The available bandwidth $bw_{AP}(vn)$ between each PC $vn$ and the AP is 3Mbps. 100 real users, 100 virtual users, and $n$ virtual objects are placed on the positions decided at random in the hybrid shared space, changing $n$ from 50 to 5000. The directions of the users are also set randomly. Each user terminal sends 60 packets of AR events every second, and the size of each packet is 32 bytes. The om-nodes are not used in this experiment. As user’s view in Fig. 4, we set the radius of a half circle to be 15m and divided the half circle equally so that the angle and the radius of each zone are $1/3$ (i.e., $\pi/3$ and 5m), respectively. We also set the importance values for zones $a$, $b$, $c$, $d$, and $e$ to 64, 16, 4, 2, and 1, respectively, since the difference between $a$ and $b$ should be larger than that among $c$, $d$, and $e$.

6.2 Required Bandwidth for User Terminals

We measured the required network bandwidth between a user node and the corresponding bwc-node under the configurations in Sect. 6.1 for both cases with and without our QoS adaptation mechanism. The average results of 100 simulations are shown in Fig. 5.

Without the QoS adaptation mechanism, the required bandwidths for each real user terminal (“RU without QoS” in Fig. 5) and each virtual user terminal (“VU without QoS” in Fig. 5) exceeded the capacity (i.e., 10Mbps and 5Mbps) when the number of objects in the whole space becomes more than 250 and 500, respectively. With our QoS adaptation mechanism, the required bandwidth is regulated below the capacity even if the number of objects increases to 5000 (“RU with QoS” and “VU with QoS” in Fig. 5).

6.3 Impact of QoS Adaptation

With our QoS adaptation method, AR events of more important objects are transmitted at higher transmission rates (i.e., framerates) than other objects, within the available bandwidth. To examine the effect of the QoS adaptation, we measured the framerates of AR events for objects in view zones $a$ to $e$ under the configurations in Sect. 6.2. The results are shown in Fig. 6 and Fig. 7.

In Fig. 6 and Fig. 7, the lines with labels $a$, $b$, $c$, $d$, and $e$ show the average framerates for objects on the corresponding zones, and the line with label uniform shows the average framerate when the bandwidth is uniformly distributed to objects.
Table 2: Subjective Evaluation by Questionnaire

<table>
<thead>
<tr>
<th>Type</th>
<th>Ips</th>
<th>ratio of right answer</th>
<th>comprehensibility</th>
<th>time to answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>With QoS</td>
<td>a</td>
<td>24</td>
<td>100%</td>
<td>4.5</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>7</td>
<td>100%</td>
<td>3.5</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>2</td>
<td>83%</td>
<td>2.0</td>
</tr>
<tr>
<td></td>
<td>d</td>
<td>4</td>
<td>100%</td>
<td>3.3</td>
</tr>
<tr>
<td></td>
<td>e</td>
<td>5</td>
<td>83%</td>
<td>3.0</td>
</tr>
<tr>
<td></td>
<td>f</td>
<td>6</td>
<td>66%</td>
<td>2.3</td>
</tr>
</tbody>
</table>

Fig. 6 shows that each virtual user can watch important objects in zones a and b at higher framerates than uniform. Especially, framerate of objects in zone a keeps more than 50 frames/sec while the number of objects is less than 4500. The framerates of less important objects on zones c, d and e are reduced below uniform.

Fig. 7 shows that each real user can also watch important objects in zones a at better framerates than uniform. However, the effect is smaller than the case for virtual users. The framerates of objects in other zones are reduced below uniform. This is because each object’s importance value is decided as the maximum value among its watchers as explained in Sect. 5.1 and large portion of objects are regarded as important objects when the number of objects is large. However, the framerates of the objects in zone a are improved to a great extent when the number of objects is less than 2000.

6.4 Evaluation by Questionnaire

We evaluated the proposed method by questionnaire based on the results above. We prepared movies in which a person draws gesture of one of the following shapes: circle, spiral, triangle heading upward, triangle heading downward, square, diamond, heart, star and of one of the following shapes: circle, spiral, triangle heading upward. We prepared a composite movie containing the three movies with the same framerate of 4fps. These framerates are decided based on the case when the number of objects is 3000, as shown in Fig. 6.

We asked five testees to see the movies and measured the time to recognize what the person in the movie is drawing, subjective comprehensibility in five levels (larger is better), and the ratio of correct answers. The results in Table 2 show that the testees recognized the important objects more accurately and quickly with our QoS adaptation method.

7. CONCLUSION

In this paper, we proposed a framework for interaction between real and virtual users in hybrid shared space, and a QoS adaptation mechanism for implementation in a network with bandwidth limitation. We confirmed that our method can handle hundreds of real and virtual users and thousands of objects with sufficient framerate in an ordinary wireless LAN and internet environment.

In this paper, we only coped with bandwidth limitation, but we are planning to extend our method to guarantee short latency. We will also enhance our method assuming that bwc-nodes send control packets to upstream so that they give feedback of QoS adaptation.

8. REFERENCES


ABSTRACT
In this paper, we propose a new middleware, which supports implementation and evaluation of Application Layer Multicast (ALM in short) protocols in real environments. The middleware provides many functions by which it can largely reduce the time and effort required for ALM protocol prototyping, performance evaluation and tuning. By using those functionalities, we have easily implemented several well-known application layer protocols and successfully performed their performance comparison in real environments. As an example, we have implemented ALMI, NARADA, NICE and OMNI as ALM protocols, and compared their performance on PlanetLab. We believe that this is the first effort made to develop this kind of ALM supporting middleware.

Categories and Subject Descriptors
C.2.2 [Network Protocols]: Applications; D.2.2 [Design Tools and Techniques]: Software libraries

General Terms
Design

Keywords
Application Layer Multicast, Middleware, Real Environment

1. INTRODUCTION
Application Level Multicast (ALM) has attracted a lot of attentions and a number of ALM protocols, each one has very good quality, have been designed for the last decade [1, 2, 3, 4, 5, 6]. However, in contrast to the success of P2P based file distribution and streaming [7], only a few ALM protocols (end host multicast protocols in particular) were actually implemented and operated [8] even though they have a great potential for launching communication or data sharing without relying on any service except the Internet infrastructure. In this paper, we address the deployment issue of ALM protocols. Our approach is to provide a middleware to facilitate to test and validate the operation and performance of a wide variety of ALM protocols in real and common environments.

Some of ALM protocols have been implemented experimentally or practically, and are ready to run on the Internet. In Ref. [2], an application level multicast communication library called ALMI has been implemented in Java. In Yoid (Your Own Internet Distribution) Project [1] wrapper scripts are provided as Yoid Software for Mbone tools such as vic [9]. End System Multicast (ESM) [10] developed by the research group at Carnegie Mellon University provides a native code toolset based on the methodology in Ref. [3]. This tool is famous for its distribution of live video in SIGCOMM2002 conference HyperCast [11] provides socket-like Java APIs based on Delaunay Triangulation methodology in Ref. [12]. RelayCast[13] is a middleware to aim at adapting to various applications that require different metrics. Our research group has also presented Java implementation of EMMA [14].

Different from these toolsets which mostly implemented specific protocols in different languages for experiment purposes, we provide a framework to design and test (existing or new) ALM protocols in real, common environments. In the sense of toolsets for overlay protocols, MACEDON [15] is a well-designed toolset which supports the development of overlay protocols. However it mainly concentrates on implementing DHT protocols and widely supports P2P overlays while our middleware is specified for the development of ALM protocols and provides more various support functions. Testing ALM protocols in real environments is quite important since their performance is strongly affected by end hosts’ capabilities and configurations, while they are highly abstracted or ignored in simulations. Moreover, the common environment enables us to build and run several ALM implementations and compare them. Performance comparison of ALM protocols by simulation has been conducted [16] and dedicated simulators have been provided [17]. However, to our best knowledge, no paper has reported the comparison of ALM protocols in real environments under common settings, and no tools and methodologies for such purpose have been presented so far. Actually, we have implemented the algorithms of ALMI [2], NARADA [3], NICE [18], OMNI [5] and compared their performance on PlanetLab to show that our middleware could help for quick and brief imple-
2. MIDDLEWARE DESIGN AND IMPLEMENTATION

2.1 Characterizing ALM applications

Existing ALM protocols can be categorized from the aspects of overlay topology (single tree, multiple tree or mesh) and management policy (centralized or decentralized). This categorization is shown in Figure 1 along with the operations and required information to perform them. After well studying these characteristics, our middleware is designed to give the maximum flexibility and functionality to all kinds of ALM protocol development.

2.2 Functionalities of Middleware

Our middleware runs on each participant node. The functionalities of the middleware are categorized to the following six categories.

2.2.1 Topology management functions

In most ALM protocols, (1) overlay topological information, (2) available bandwidth of nodes, (3) connectivity and (4) delay between each pair of nodes are utilized. For instance, newly joining nodes are favorably connected close to the source node as to satisfy the delay constraint in a multimedia streaming application. These functions are described in the rest of this section, after categorizing them into, (a) common functions, (b) centralized protocol specific functions and (c) other protocol specific functions.

The common functions provide the following information, which can be used by all protocol types of Figure 1. As node’s information, node’s available bandwidth (dedicated LAN bandwidth), utilized bandwidth ((h) and (i) in Figure 1) and any other protocol-specific information are provided. In addition, IDs and network addresses of neighboring nodes ((j) in Figure 1) are provided as node’s connectivity information. Finally, overlay and physical network delay to any other node ((k) and (l) in Figure 1) is provided as delay information. This information is stored in each node in a decentralized scheme and in the administrative node in a centralized scheme.

The centralized protocol specific functions are common for all types of centralized ALM protocols. Their corresponding information includes: IDs and network addresses of all participant nodes ((a) in Figure 1) as node set information, physical network delay and bandwidth between each pair of nodes ((b) and (c) in Figure 1) as node pair information, and join, leave timestamps and communication history as node log information. This information is stored in the administrative node.

Finally the other protocol specific functions provide the following information for centralized single-tree, multipletree and mesh protocols: root node(s) and tree structure(s) information ((d), (e) and (f) in Figure 1) for single-tree and multiple-tree protocols, the maximum delay and hop count of tree and those from the root node, and mesh structure information for mesh protocols ((g) in Figure 1). This information is stored in the administrative node.

2.2.2 Basic communication support functions

Our middleware provides the following functions for basic communication purposes; establishing/destroying overlay links between nodes, message passing (send, receive and forward), and task scheduling for specified timestamps.

2.2.3 Multimedia support functions

Streaming protocols with their individual implementations cannot be simply compared as the streaming performance largely depends on the implementation method. So this functionality of our middleware largely benefits the ALM application development; multimedia passing (send, receive and forward), and real-time play backing of the received multimedia stream (under development). Here, multimedia data passing is done using RTP/RTCP.

2.2.4 Performance monitoring functions

The following performance monitoring functions become quite important for the maintenance of developed ALM applications: real-time displaying of node connectivity, bandwidth, delay, packet-loss and jitter, real-time identification of bottleneck nodes, and logging of messages, multimedia passing history and overlay link establishing/destroying history. The information required to realize these functions is acquired from the topology management functions, the basic communication support functions and multimedia support functions.

2.2.5 Execution support on PlanetLab

This functionality largely reduces the efforts required so far for carrying out large scale experiments on PlanetLab; remote manipulation of nodes such as adding, removing and replacing, and node task (join, leave, etc) scheduling.
2.2.6 ALM protocol design support functions

These functions are provided for developers as high-level (algorithm level) APIs: (i) basic tree construction: each node is greedily connected to the tree satisfying a given condition (e.g., minimum delay). (ii) k-connected mesh construction: k neighbors are randomly selected, (iii) basic tree recovering: recovering of the tree in a single node absence (leaving or failure), by connecting the disconnected subtree to left-node’s parent, (iv) information collection: periodic collection of information, such as neighbor node information, tree/mesh diameter or depth and (v) node swapping: child-parent, child-grand parent, sibling wise, child wise, random swapping of nodes satisfying a protocol-specified condition (e.g., delay).

3. IMPLEMENTATION OF ALM PROTOCOLS

In this section, we explain implementation of some well-known ALM protocols, using our middleware. The complete list of classes and APIs of our middleware will be given in [19]. In each of the following examples, we first outline their protocol behavior and then describe how our middleware helps the implementation of those protocols.

3.1 ALMI

ALMI is a centralized, tree-based protocol. It mainly consists of tree construction procedure and tree refining procedure. As for the joining procedure new nodes are required to inform their joining ambition to the tree-controlling node. Then the tree-controlling node decides and informs appropriate positions for those nodes. Finally, the new nodes have the responsibility of positioning themselves following the controlling node’s instructions\(^1\). Our middleware implements this join procedure as shown in the pseudo-code of Figure 2. Note that the code only outlines the behavior of the protocol, omitting the verbose parts like error handling.

A newly joining node sends ALMIJoinRequestPacket to the SuperNode that controls the entire tree. The middleware provides a handler called onReceiveALMPacket, which is executed on receiving a message at a node. Therefore the behavior of SuperNode for ALMIJoinRequestPacket can be described in onReceiveALMPacket handler (line 02), such as, (a) finding the connection situation of the tree by calling CentralizedTreeConnection (line 03), (b) getting the list of nodes where the residual degree is not zero (line 04), (c) randomly selecting one among them, and, (d) informing the selected candidate to the joining node by sending ALMIJoinReplyPacket.

In ALMI, periodical tree-refining procedures are carried out at SuperNode after the initial construction. The middleware supports this by providing onTopologyUpdate handler, which is periodically executed. The actual refining is done by making the nodes change their parents, and our middleware’s ChangeParentPacket can be used to inform the new parent candidates to the corresponding nodes (lines 16 and 20). Basically, the grand parent is selected as the new parent candidate, and the node with the minimum delay is used in case the grand parent has no residual degree (lines 15 and 18).

\(^1\)Note that these basic steps are typical for most centralized join procedures.

Figure 2: main part in source code of ALMI

3.2 OMNI

OMNI is also a tree based ALM protocol. But it is different from ALMI in the sense of distributed tree management. However, considering the fact that onReceiveALMPacket handler is waiting at each node, join requests from multiple nodes can be handled simultaneously at different nodes. So the behavior on receiving OMNIJoinRequestPacket at each node can be described in onReceiveALMPacket handler (lines 02–03 in Figure 3). Here, the joining is permitted by replying the join requesting node with OMNIJoinReplyPacket, in case the adopting node has residual degrees (lines 03–05). Otherwise, delays to the current neighbors are measured (line 08) after queuing and then the reply to N\(_{\text{new}}\) is done depending on these delay values.

Note that it takes different measuring-times because of different network connectivity, and the reply should be done once the delay values of all neighbors are collected. For this purpose, our middleware provides another event handler called onEvent which is executed when the delay measurement is complete (lines 12–13). Therefore, the behavior for the corresponding event occurrence can be described in onEvent handler as follows. OMNIJoinReplyPacket is sent to N\(_{\text{new}}\) (line 21) if it has a smaller delay than N\(_{\text{max}}\), which is the neighbor with the maximum delay (line 18,19). In the same time OMNIChangeParentPacket is sent to N\(_{\text{new}}\) (line 20). These two messages make N\(_{\text{new}}\) replace N\(_{\text{max}}\), and N\(_{\text{max}}\) move to N\(_{\text{new}}\)’s child position. If N\(_{\text{new}}\) has no smaller delay than those of the current neighbors, the connection is refused by an OMNIJoinRejectPacket (line 23).

This example illustrates how easily our middleware can describe the behaviors of ALM protocols, even though they consist of asynchronous procedures specific for decentralized
3.3 NARADA

NARADA is a mesh-based decentralized protocol. Here, the newly joining node $i$ becomes a member of the mesh by connecting itself to the node sub-set $N = \{n_1, ..., n_k\}$. The entire node list is assumed to be known by some application.

Similar to ALMI, NARADA also consists of a periodical refine procedure. However, this differs from that of ALMI in the sense of decentralized execution. In NARADA, the delays to non-neighbor nodes are periodically measured and new connections are established to those with less delay. Remembering the decentralized join procedure of OMNI, it is clear that this delay measurement and corresponding refining procedure. However, this differs from that of ALMI in the sense of its hierarchical structure. NICE constructs multiple clusters where each one forms a small mesh, and it constructs another overlay mesh connecting those clusters to form a hierarchical mesh. As we already mentioned, our middleware provides the basic classes, BasicCentralizedTopology, BasicDistributedTopology, BasicCentralizedTreeTopology and BasicDistributedTreeTopology, to support centralized / decentralized, tree-based/mesh-based topologies. However, in hierarchical-decentralized mesh topologies like NICE, the management of each cluster and their upper layer mesh, should be done separately.

For this purpose the middleware provides a new class called LayeredDistributedTopology extending BasicDistributedTopology, the basic distributed mesh-topology management class. As the basic mesh-topology can be controlled by our middleware's BasicDistributedTopology class, (1) leaderNodeId: the leader node of cluster, (2) nextLeaderNodeIds: the leader of neighboring clusters, (3) nextLayerNodeIds: nodes of lower layer clusters, and, (4) methods to set/get the values of (1)-(3) variables, are required to be added. The code for (1)-(3) is shown in Figure 5. The code for (4) is omitted due to the space limitation.

Figure 6 summarizes the lines of code (LOC) counts for each protocol explained above. All protocols are easily implemented in a few hundred LOCs. This result shows the usability of the middleware. It reduces the work of implementing basic communication functions and multimedia functions. In addition, the functions specialized for ALM protocols largely help the implementation.
4. PERFORMANCE EVALUATION

We compared the performance of ALMI, OMNI, NARADA and NICE, the ALM schemes implemented with the middleware. The experiments were carried out on PlanetLab using our middleware’s execution support function, and the multimedia support and performance monitoring functions were used for the evaluation.

4.1 Experiment environment

First, we describe our experimental environment.

- number of terminals: about 100 (Asia:10, Europe:40, America:45 and some others)
- terminal configuration: Pentium3 (1.2GHz) – Pentium4 (3.4GHz), 512MB – 3.6GB memory, Linux OS version 2.6.12-1.1398.FC4.5.planetlab
- Java: JDK1.5
- RTT (round trip time) of each pair of PlanetLab nodes: mean 150ms, max 14000ms

4.2 Scenario of experiments

Overlay delay and RDP (Relative Delay Penalty) were selected as general evaluation metrics and jitter and streaming bandwidth were selected as the streaming evaluation metrics.

The scenario of our experiments is as follows. First, 20 nodes joined the application forming the initial topology. Then streaming was started after assigning the “center node” as the streaming-source. Here, the node joined first was considered as the center node and the streaming-rate was set to 500kbps, where we assume a video streaming application. Note that we selected the same node as the source node in each protocol.

Next, another 20 nodes were added and the same experiment was carried out, and this was repeated until the number of nodes reached 100.

4.3 Experimental results

The experimental results are shown in Figures 7, 8, 9 and 10. Figure 7 shows the average streaming delay (network delay + node delay) of each ALM scheme, and it states that NARADA and NICE, the mesh-based protocols, have smaller delay than ALMI and OMNI, the tree-based protocols. Especially, OMNI’s delay is much larger than the others. One possible reason for this is OMNI’s defect of performing its refinement only looking at the network delay. So it may select nodes with small link delay and large node delay, which results in high streaming delay. However the performance of these schemes from the aspect of link delay is presented with RDP metric in Figure 8. We can say that the overlay delay is rather large in NARADA. Note that we defined RDP as the ratio of ALM protocol delay (including data buffering delay at intermediate end hosts) to the unicast delay which was measured by ICMP echo packets. Figure 9 also says that the mesh protocols have smaller jitters, which leads to better streaming performance. On the other hand, Figure 10 states that the mesh protocols have bigger bandwidth consumptions.

As it is already clear to the readers from the above experiments, our middleware well supports the performance comparison of variety of ALM protocols. Also as the environment-dependent (thus unexpected) behavior of protocols, we have experienced large jitters in OMNI and ALMI from around 80 nodes. The protocol developers can improve, customize and tune their implementation in real environments by considering those protocols’ strong and week features known by their nature of design or by simulations as well as site-specific and application-specific behavior observed using our middleware.

5. CONCLUSION

In this paper, we proposed a new middleware for supporting design and implementation of ALM protocols in real environments. Here, we have implemented ALMI, NARADA, NICE and OMNI using our middleware, and carried out
their performance comparison on PlanetLab. The comparison can be done in a few days, and several comparison results have been obtained using our middleware (we omitted some details of the comparison due to the space limitation). We have a plan to open our middleware from on WWW soon [19]. More detailed comparison with several ALM protocols in real environments is part of our future work.

6. REFERENCES

For seventeen years, NOSSDAV has fostered cutting-edge, state-of-the-art research in multimedia and newly emerging areas such as networked games and peer-to-peer streaming. The workshop environment encourages lively discussion among participants and invites strong feedback for work in progress. In 2007, NOSSDAV will be held at the University of Illinois at Urbana-Champaign IL, USA. UIUC is in driving distance to Chicago, offering spectacular architecture, art museums and live jazz. Urbana is close to various national historical landmarks including President Lincoln’s house and library.

NOSSDAV invites submissions on all areas of multimedia computing and networking and strongly encourages work in progress in emerging areas. Papers grounded in high-quality experimental research based on prototype and real systems are highly valued. NOSSDAV will give extra consideration to papers that aim to comprehensively validate previous work in some topics within multimedia, propose new directions for research or call into question existing conventional wisdom. Topics of interest include, but are not limited to:

- Peer-to-peer streaming
- Networked games
- Sensor networks and architectures
- In-network stream processing
- Wireless and mobile multimedia systems
- 3D multimedia and tele-immersion
- Streaming 3D graphics and virtual worlds
- Application-level multicast
- Multimedia security
- Digital rights management
- Real-time operating system support for multimedia
- Multimedia middleware and frameworks
- Multimedia grids
- Programmable and GPU/SPU-enabled multimedia

A broad view will be taken in deciding what topics are within scope. Please feel free to contact the workshop co-chairs if you are unsure and wish to check if a particular paper or topic is within the workshop scope. As always, student participation is strongly encouraged. To encourage a good mix of seasoned researchers as well as students, we will be offering discounted registration for non-student co-authors of a paper or other non-student participants associated with the author of a paper (participants from the same institution). Submissions (as well as the camera-ready final versions of accepted papers) should be no longer than 6 pages. We expect these submissions to be the kernel of what will eventually lead to full-length papers at high-quality conferences or journals.

Co-chairs: Reza Rejaie (University of Oregon), Klara Nahrstedt (UIUC)

Important Dates:
- Paper registration: February 5, 2007
- Paper submission: February 12, 2007
- Notification: March 31, 2007
- Camera ready: April 19, 2007
- Workshop: June 4-5, 2007